

AD-A146 773

LOCAL AREA NETWORK EVALUATION BY MICROSS EXTENSIONS(U)  
POLYTECHNIC OF CENTRAL LONDON (ENGLAND) Y PAKER FEB 84  
DA-ERO-78-G-110

1/1

UNCLASSIFIED

F/G 9/2

NL

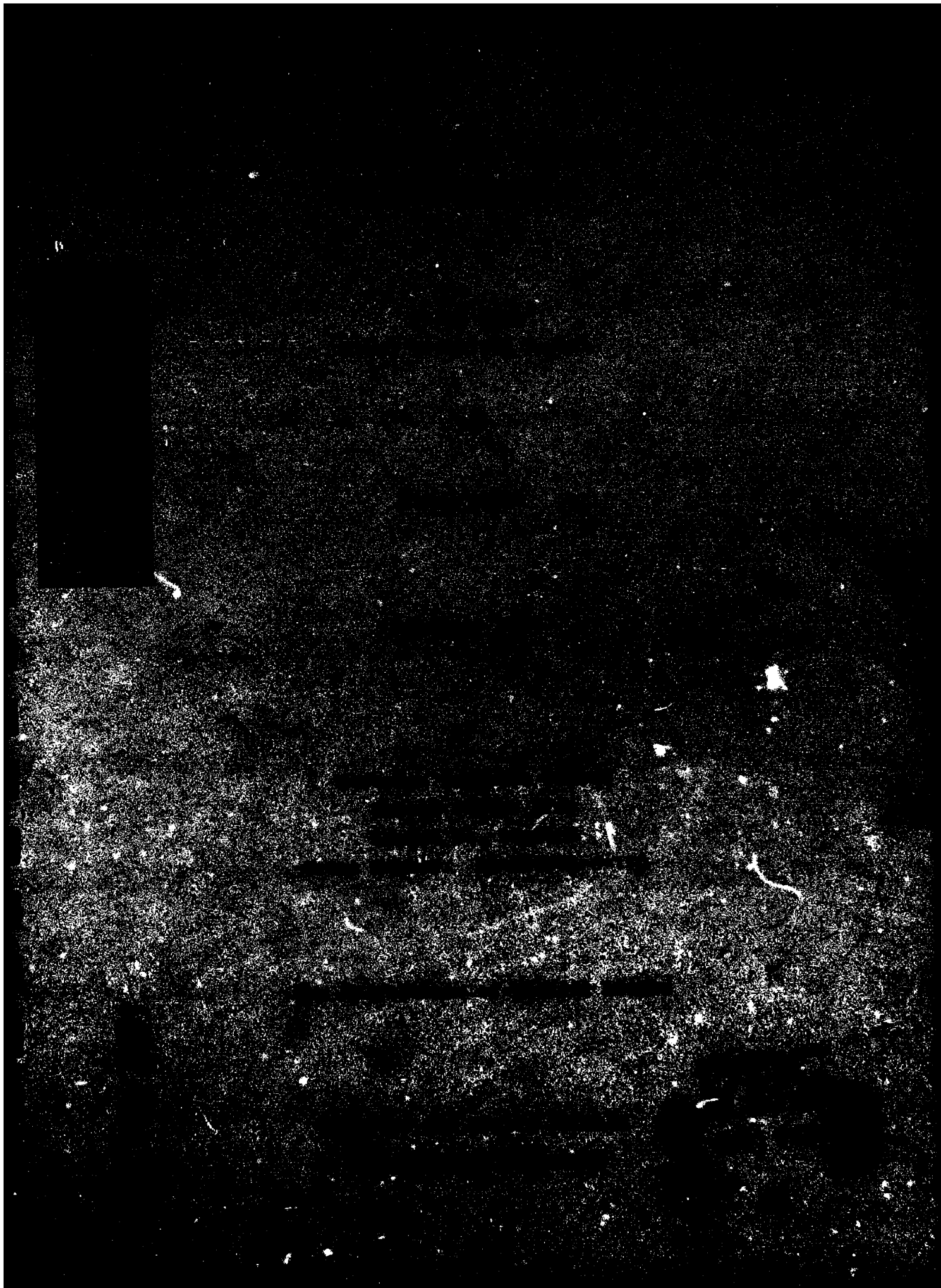


**END**

**FILMED**

**11-84**

**DTIC**



LOCAL AREA NETWORK EVALUATION  
BY MICROSS EXTENSIONS

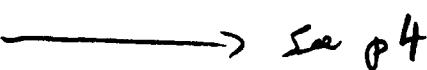
Y. Paker

<u>Contents</u>	<u>Page</u>
1. Introduction	1
2. Modelling and Simulation of a CSMA/CD Contention Bus	5
3. Aspects of Voice Communications on the Local Area Network, Ethernet.	24
4. Modelling and Simulation of a Token Passing Ring (Bus)	40
5. Modelling and Simulation of a Circulating Slot Ring	59
References	73
Appendix: A Sample Run for CSMA/CD Contention Bus Simulation	78

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



## 1. INTRODUCTION

This report reflects the work done during the third year of the three year project on Control, Synchronisation and Fault-Tolerant Operations in Variable Topology Multicomputer Systems. The first year has been reported in [1]. The overall purpose of the research work has been to investigate the architecture of a network of low cost computers (mini- or microcomputers) linked with serial communication paths which can be reconfigured according to the needs of each computation. This led to a novel architecture named Variable Topology Multicomputer, or VTM for short, developed by a previous three year US European Office Grant [2], [3]. The second year's work has undertaken the evaluation of a number of interconnection topologies [4]. 

The construction of parallel computer structures where relatively large numbers of computers, reaching two or three digit size, has become feasible due to the advances in VLSI technology which led to very low cost microprocessors, memories, and ancillary circuits. Minicomputers have already introduced some degree of de-centralisation in real time systems where it has become possible to place a computational resource nearer to the point where such a requirement exists. Such loosely linked computer networks are particularly attractive where the computational task is spatially distributed and has to be performed in real time. Microcomputers have accelerated this tendency to a greater extent by offering

the designer remarkably flexible, powerful and low cost components. Microelectronics industry has produced more powerful circuits such as 16-bit processors and 32-bit processors currently being introduced. On the high performance side, the US Defense Department project is an effort to achieve very high performance microprocessors [5]. Thus the technology limitations for a microcomputer as a point computing resource is yet to be reached. The availability of high performance yet low cost microprocessors is making the use of aggregates of such components as a tightly coupled structure more and more attractive. The software support available for more powerful microprocessors has also greatly improved; in addition to having high level language compilers e.g. Fortran and Pascal, sophisticated operating systems such as Unix are available for these machines. This provides a much richer environment for developing multi-microprocessor systems [6].

Another significant development has been the emergence of Local Area Networks (LAN's). These provide an interconnection medium for relatively large numbers of computers (mainly mini or micro). The contention bus scheme of Ethernet [7], the circulating slot structure of the Cambridge Ring [8] and token passing ring [9] are some of the well-known approaches. These are fixed topology architectures yet they offer fairly standardised interconnection media, very different from the store-forward scheme used for long-haul networks. The standardisation is now being extended to the communication protocol level which goes beyond the link level protocols by IEEE

802 standards work.

Many of the underlying control and synchronisation problems are receiving greater attention, some due to LANs, others due to the greater use of distributed computing [10]. Distributed Operating Systems are of much interest as are the higher level language constructs for synchronisation and control, for example the rendezvous concept of the Ada language is designed to achieve synchronisation between co-operating tasks.

During 1979-81, a distributed computing simulation package called MICROSS was designed and developed by Dr. Bozyigit and this author, supported by a Science and Engineering Research Council research grant. The details of this package are introduced in [4]. This system enables a designer to enter the specifications of a multi-processor system in terms of topology, transmission line and processor characteristics in an interactive manner. Following a simulation phase the reporting is done by plotting performance figures such as delay times, throughput, etc. Thus MICROSS is a powerful package to evaluate various architectural approaches and topologies graphically in an interactive manner. MICROSS facilities are now offered as a service to various research groups and laboratories who would like to test the design options that they may have in distributed systems. MICROSS is seen as an important tool to evaluate a given architecture against a given application requirement.



→ The third, and the final, year of this work concentrated on extending MICROSS to be able to model LAN's. MICROSS implements, in principle, store-forward networks. To accommodate other types of connection schemes, namely contention bus, token passing ring (bus), and circulating slot, we needed means of handling such structures. Over the period of this project, LAN's have become more important providing standard means of interconnecting computers up to a size of a few hundred machines. Currently work is in progress to enable building more complex structures of interconnected LAN's such as the Cambridge Fast Ring [11]. Therefore, it seemed important to extend MICROSS to be able to model such novel architectures. Furthermore, in the modelling work we moved to a more total service environment and made some studies of voice transmission over such networks.

In this respect, three major LAN types have been modelled. Section 2 presents modelling and simulation of a CSMA/CD contention bus. Section 3 uses this model to investigate voice communications on Ethernet. Section 4 investigates token passing ring modelling. Section 5 considers the modelling of an empty slot ring.

To the work reported here T.O. Brunner, H. English and Dr. M. Bozyigit made contributions.



## 2. MODELLING AND SIMULATION OF A CSMA/CD CONTENTION BUS

### 2.1 Summary

This Section describes a simulator for a general CSMA/CD contention bus and Ethernet. It was created as an extension to the MICROSS package developed at the Polytechnic of Central London. The simulator will generate classes of packetised traffic with various statistical characteristics, and record collision statistics, average throughput and the packet delay distribution. Thus it is a tool for evaluating the performance of a local network; in particular it supports the investigation of system degradation due to packet delay. A number of simulation runs have been completed and results compared with theoretical predictions.

### 2.2 Description of CSMA/CD Contention Buses

Over the last fifteen years, theoretical and practical developments have led to the current Carrier Sense Multiple random Access with Collision Detection protocol. Several requirements have shaped it:

- (1) that there be one shared broadcast channel
- (2) that the access technique use that channel for scheduling as well as transmission, and
- (3) that the access technique be fully distributed.

The further choice of highest performance at low channel utilisation (appropriate for terminals connected to a host computer, for example) and conceptual simplicity led to the contention bus.

The seminal implementation was the ALOHA packet radio network [12], [13], [14]. Here the various stations, while constantly listening to the channel, broadcast when they have information to send, perhaps colliding with each other, and repeat the process if their message doesn't get through to the destination. A key feature is the randomised delay before retransmission. If two stations broadcast at the same time, their packets will collide. After receiving no acknowledgement, the stations will retransmit at the same time - since they follow the same algorithm - causing another collision. It is necessary to break this lockstep so a randomised time delay before retransmission is introduced.

If the station listens to the channel - senses carrier - before transmitting, and only broadcasts when the channel is quiet, performance is improved. The number of collisions is reduced. Further, it is possible to design coaxial transceivers so that they sense the channel as they are broadcasting and thereby determine if there has been a collision as it happens. Responsibility for the detection of a collision is shifted to a lower level of the protocol, away from the acknowledgement/time out feature used in ALOHA. The responsibility for retransmission

is shifted also. Once the transceivers have detected a collision, the two stations immediately abort their transmission, and broadcast a short jam sequence so all stations will recognise that a collision has occurred. Even if the jam sequence experiences a collision it will still be recognised. The two reschedule transmission, introducing a random time delay. In the CSMA/CD protocol the only way a collision can occur is if a second station starts transmitting so soon after a first that the first's broadcast has not had time to propagate down the cable to be sensed by the second. If no second broadcast has started within the end-to-end propagation delay, the first station has seized the channel and will have a successful transmission.

Several algorithms for determining the length of the retransmission delay have been presented and the performance of the resulting system has been analysed [15], [16]. Ethernet [17] uses an algorithm known as the truncated binary backoff which produces a delay whose mean increases geometrically with the number of collisions. These algorithms' primary purpose is to ensure stability of the protocol [18], [19]. One result of their behaviour is that multiple collisions result in sizeable delays; Ethernet throws away packets, after fifteen collisions, as undeliverable.

There have been three different versions of Ethernet [17], [20], [21], and this simulator will properly model all three.

## 2.3 Simulation Model

Our simulation program, like the MICROSS [22] and NPL-Multi-computer Ring Simulator [23], is menu driven; there are three significant subroutines that may be called by the user in any order. They are: system definition (1), system simulation (2) and system reporting (3). The menu also offers two choices restart (4) and snapshot (5), which are not implemented, and exit (6).

### 2.3.1 System Definition

The definition routine zeros all data storage areas, erasing all previously stored information, initiates the data structures so a simulation can begin, and asks the user to define the system (refer to Appendix I to see a sample run).

#### Select Hardware

Channel capacity and cable length are specified. The Xerox experimental Ethernet used 3 M bits/sec and 1 km. The user is free to choose any desired values [20].

#### Select Protocol

A choice of Ethernet standard and a generic CSMA/CD is offered. The differences between the two are given overleaf.

	CSMA/CD	Ethernet
Maximum data length (bits)	16000	12000
Minimum data length (bits)	0	368
Header (bits)	24	208
Number of collisions before abort	250	15
Minimum packet spacing (m sec)	0	9.6
Jamming period (bits)	0	48
Bit recognition (m sec)	0	0.1
Binary backoff's slot time	round trip delay	512 (bits)

#### Define Topology

The total number of nodes is currently limited to 256. Of these, any number could be data or voice nodes. The difference between the two lies in the statistical character of the traffic they generate. The two traffic classes are to be used by themselves, or together to investigate the results of traffic integration.

#### Define Traffic

For data traffic the inter-arrival time (IAT) may be fixed (constant), or exponentially distributed (Poisson arrivals). The user chooses the value of the IAT, or the mean of the exponential distribution, as appropriate. The packet length also may be fixed or exponentially distributed, and the user similarly chooses the length or the mean of the length distribution.

For voice traffic, the IAT may be fixed or it may correspond to voice traffic which has been submitted to speech activity detection before packetisation. In this case, packets are sent regularly while there is speech and nothing is sent during silences, so the IAT is fixed during the (exponentially distributed) talkspurt and is then exponentially distributed reflecting one silence period. The cycle of talkspurt and silence is repeated. If the user has chosen speech activity detection, the choice of means for the talkspurt and silence distribution must be made. Given the exponential approximation, we would suggest means of 1.3 and 1.7 sec respectively [24]. Finally, the user chooses the number of data bits in each packet. In the absence of speech activity detection, the ratio, packet size over IAT, yields the bit rate for a voice node. Data compression through coding can be modelled here, by reducing the packet size. Corresponding processing delay can be modelled in the reporting routine.

#### 2.3.2 System Simulation

The Simulation Routine proceeds for the duration of the user's specification (SIMT) and adds the statistics it gathers into the data storage areas. Furthermore, it continues where it had left off, so that the only difference between many short simulations and one long one is the way seeds for the random numbers are handled.

### Random Numbers

The simulator randomises all the seeds used by the pseudo-random number generator (FORTRAN'S RAN), at the beginning of the simulation routine. They are read in from a file (SEEDS. RND), each randomised by another pseudo-random number generator (IRAND), and these new values are used as seeds for the random values generated during the simulation: IAT, talkspurt, silence, packet length, rescheduling delay.

### Event Generation

A new packet, arriving at the host for transmission on the channel, is called an event, and is discrete. The routine to generate an event (GENERA), mathematically shapes the random values that characterise the event in accordance with the probability distribution of the host's traffic. The IAT and packet length could be constant or exponentially distributed with different means, for example. The pertinent values for the event are stored both on the common block for host and channel description (RNGHWR) and in the one for packet description (PACKET).

Only after a host's previous packet has been successfully transmitted or lost, is this routine called to generate a new event. Only one event per host at a time is kept on the event queue; other events are not generated till needed. Thus a

particular host's queue length is understood to be the difference between the time its last untransmitted packet arrived (SYSENT) and current simulation time (TIME). The average queue occupancy can be calculated from Little's formula [25]:

av. Q length = av. Q delay x arrival rate.

### Event Scheduling

After generation, events are placed on a queue and ordered by their scheduled time for transmission (SCDTIM). The routine to accomplish this (SCHEDU) ensures that the top of the event queue (NEXTAC points to it), is the next packet that will start broadcasting on the channel (if it is quiet). There is also a queue of unused packets, the available queue (NEXTAV points to it), and together they are the common block for events (PACKET).

### Simultaneity

Collision for CSMA/CD style protocols, occurs when two stations start broadcasting without enough time for the first station to be heard by the second. At the most, this time is the end-to-end trip delay plus the bit recognition time, a sort of window of vulnerability. The simulation always assumes the worst case, that if two stations transmit within that window they will collide, as if they were on opposite ends of the transmission cable. From another point of view, the two events are considered



simultaneous. The time increment, or step size, of the simulation(SIMINC) varies, but in the case above will be the window. The simulation considers the two events simultaneous because they fall within the smallest time resolution (i.e. the step).

#### State of System

The simulator is state driven in that the channel's state (IRSWCH where 0: no transmission, 1: attempting transmission, 2: successful seizure, 3: collision, 4: packet spacing) determines the step size and the action of the simulator during any loop cycle (Figure 2.1). So the simulator is continually cycling through a sequence of states that constitute the access protocol (Figure 2.2). When there are no packets to be transmitted (state 0), the simulator merely increments time up to the next event. With at least one event present (state 1), the simulator checks if a second event occurred within the collision window by counting the number of simultaneous events in its simulation step. If no collision occurred (state 2), the simulator increments time up to the end of the packet's transmission and records that the transmission was successful. It also defers any other packet's transmission attempt by rescheduling it after the completion of the first's transmission. Finally, the completed event is removed from the top of the event queue, and the empty data structure is returned to the available queue. If there had been a collision detected (in state 1), the

colliding packets would be rescheduled by a routine that implements the binary backoff algorithm (BBOSCH). SCDTIM is changed, but SYSENT is not. So the event is repositioned in the event queue, and SCDTIM less SYSENT yields the time the packet spent waiting till it could successfully transmit. Once a collision is detected (state 3), the simulator increments time by the amount necessary for a collision to be detected and for each station to jam the channel. All events from this time increment are deferred. (Note that collision rescheduling is done in state 1 not state 3. The reason is that events scheduled for transmission during the jamming period are not distinguished from those of the collision window, once the simulator is in state 3.). Lastly, the simulator continues to defer all events which occur during the mandatory spacing between packets (state 4). It is the case that the simulator will operate correctly if the spacing period is set to zero at system definition. Similarly, the time increment for the quiescent state (0) will sometimes be zero, however, the other three states have fixed increments.

### 2.3.3 System Reporting

System reporting does not modify data storage areas. So it may be used in between calls to the simulation routine in order to provide a 'running total' report of statistics gathered.

The first part of the report presents two key statistics: average transmission delay and average throughput. Together they can be used to trace out a throughput/time delay curve of the kind commonly used to present the performance of an access protocol. The average queueing delay is also shown; it differs from the transmission delay in that it only includes time spent waiting in queue and not time to transmit the packet.

Also shown are the number of bits transmitted, packets transmitted and packets lost from too many collisions. From these last two statistics one can get an idea of whether the transmission has run 'long enough', and the amount of congestion. If processing delay needs to be included in the simulation, it can be directly added on to the average transmission delay.

The second part of the report is optional, and details, at each queue, the arrival time of the event not yet transmitted. This instantaneous servicing delay gives an estimate of the instantaneous queue length via the arrival rate. It is useful as another measure of congestion, particularly if this reporting routine is used several times during a complete simulation run.

The third part of the report is optional. It presents part or all of the queueing delay and transmission delay distributions from which the average values in the first part of the report is derived. For applications where timing considerations are

important, voice transmission for example, these distributions will indicate the number of packets delayed beyond some cut-off value, the number that will fail to reach their destination on time. These values cannot be deduced from measurements of averages.

The fourth part of the report is also optional. In it the delay distribution is broken down by the number of collisions a packet experiences before successful transmission. Because collisions take up so much time (detection, jamming, rescheduling), this information could prove useful on constructing modifications of Ethernet for time sensitive applications. Two measures of system congestion are shown: the number of attempts to seize the channel, per unit time, and the average number of attempts per packet.

#### 2.4 Comparison of Simulation with Analytic Results

Numerous simulation runs have been completed using the data traffic facilities of the simulator. The first were compared to analytic curves in order to verify the simulator's accuracy. As confidence increased, studies were undertaken with the voice traffic facilities, and are reported in Section 3. This section details the results of the original comparisons.

The work of Lam [16] will be used as the analytic model to which

we compare our simulator. It assumes a CSMA/CD protocol, one which is identical to our CSMA/CD model in the following respects. Collision and detection are assumed to occur within the same time slot and abortion of transmission to be immediate. For that work a time slot is a round trip, end-to-end transmission delay. Our (CSMA/CD) simulator sets detection and jamming time to be zero, and assumes the worst case - that signals take the full round trip delay to propagate. Thus our propagation, detection and jamming cycle takes one time slot also.

However, the analytic model makes an assumption about the rescheduling algorithm that need not always match our simulator. It assumes that the algorithm will appropriately choose the delay between collision and retransmission attempt to ensure a constant probability of a successful retransmission. This means the delay distribution is geometrically distributed and independent of both the congestion and the distributed queue's length. Thus the algorithm is assumed non-adaptive, yet unaffected by congestion. The result is an assumption that the system is always stable (and a tractable analytic formulation). However, for the purposes of our comparison we may note that this stability assumption requires any temporary period of congestion - many stations colliding and the distributed queue's length building up - to be short compared to the retransmission delay. The binary backoff algorithm [9] for our (CSMA/CD) simulator uses a slot time that follows the literature, one round trip delay, but

won't choose a delay of more than a few slots until several collisions have occurred. Any congestion over a time period of more than a few slots, that is more than a few consecutive collisions, will violate the stability assumption of the analytic model. It seems likely that under conditions of large packet size and large packet delay the assumption will sometimes be violated. If so, the simulation may yield higher delays than the analytic curve, since packets would be rescheduled into periods of high congestion and face further collisions. This divergence of simulation and analytic prediction is observed, and we believe the foregoing to be the most significant factor leading to it.

The throughput/time delay curves have been reproduced from a paper on local-area subnetworks [26] which show CSMA/CD performance in different parameter ranges (see Figure 2.3 and Figure 2.4). The ranges are low and high alpha [16], and correspond to delay asymptotes at approximately 90 per cent and 60 per cent of maximum throughput respectively. Above alpha is defined as

Alpha = Carrier sense time/mean message transmission time.

Simulation points from both the Ethernet and the CSMA/CD simulators are superimposed on the curves. The CSMA/CD simulation points essentially match the curve on Figure 2.4, while on Figure 2.3 they report higher delays on the region of high congestion. As mentioned above, we believe the explanation for this disagreement is to be found in a

simplifying assumption of stability in the analytic model, and that our simulation points are a more accurate guide to system behaviour.

The Ethernet simulation points do not match either curve precisely; Ethernet is a different protocol than CSMA/CD, one that includes practical considerations. The five points of difference are that Ethernet includes 208 bits of header instead of 24 and has stricter limits on its maximum and minimum packet sizes. As a result of the larger header, the average total packet size is larger and the time delay normalised over the (larger) transmission time is less. The result is simulation points shifted below the analytic curve. The next points of difference are the detection and jamming delays, and the minimum packet spacing in Ethernet: collision detection and recovery takes longer. Further, the reschedule algorithm uses a much longer slot time (ten times in Figure 2.3 and Figure 2.4) yielding even greater collision delays. The result is simulation points shifted above the analytic curve in the region of high congestion. The difference is pronounced in Figure 2.3, where in spite of the overall shift below the analytic curve, in the high congestion region, the simulation points surpass the analytic curve. In Figure 2.4, they merely regain the curve in the high congestion region.

```
repeat until (end of simulation)
  begin

    if (any station without a packet)
      then call GENERA

    case (state of system)
      begin
        0
        1
        2 } choose step size
        3
        4
      end

    count simultaneous events

    case (state of system)
      begin
        0
        1 collision detection
        2 record successful transmission
        3 reschedule
        4 defer transmission
      end

    end.
```

Figure 2.1 - MAIN SIMULATION LOOP



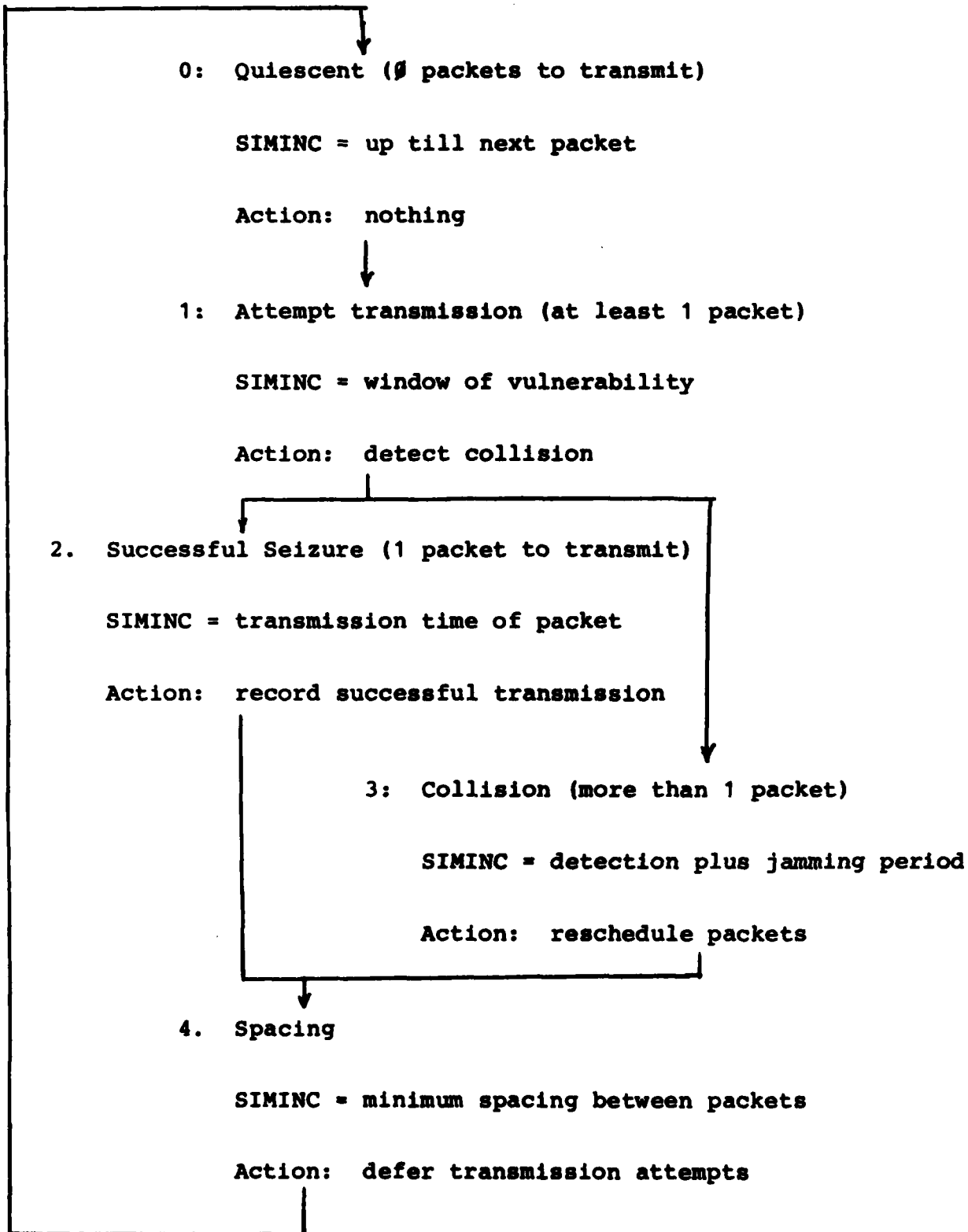


Figure 2.2 - STATE TIMING DIAGRAM

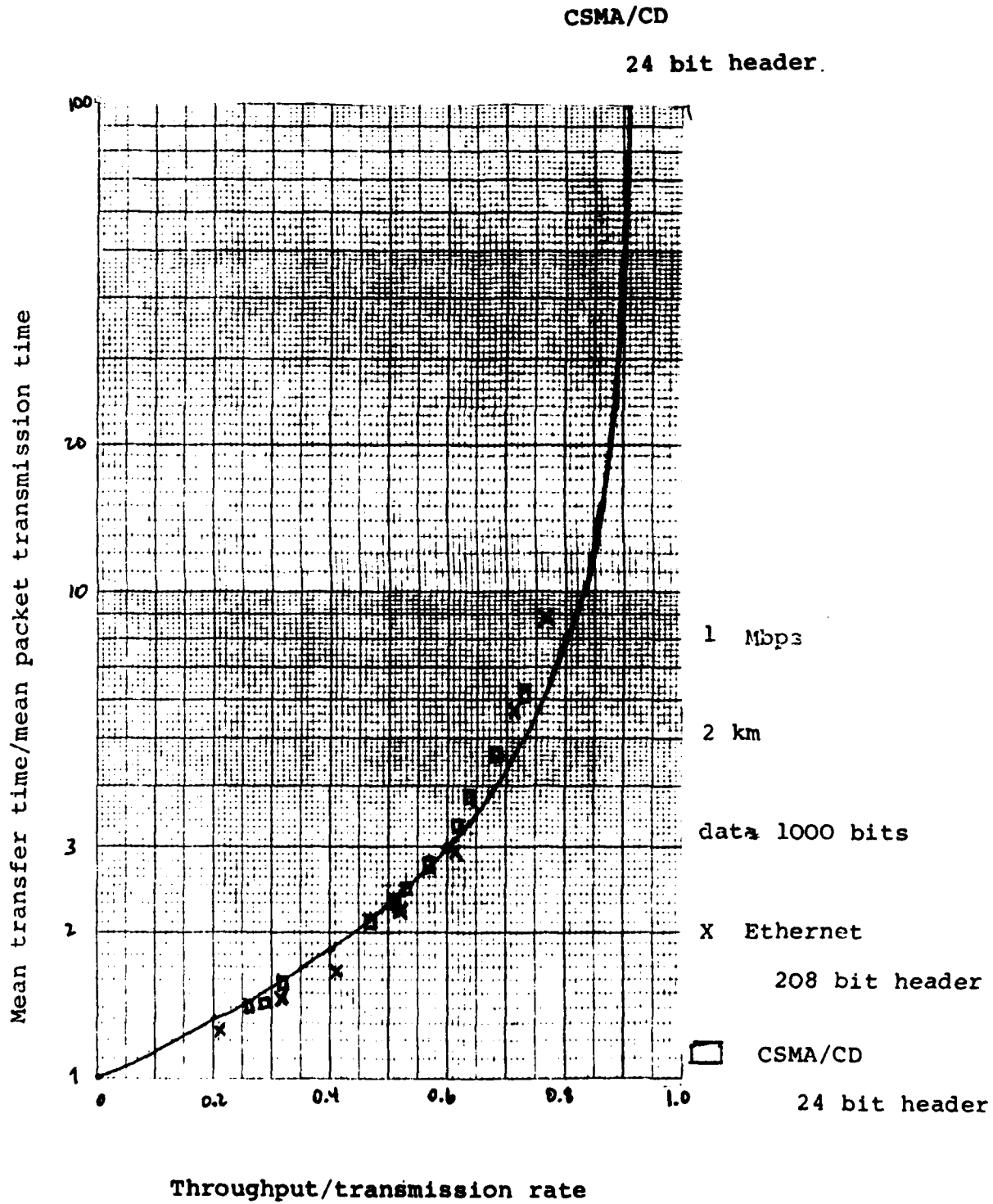


Figure 2.3 - THROUGHPUT/TIME DELAY CURVE FOR  $\alpha = .0088$

CSMA/CD

24 bit header

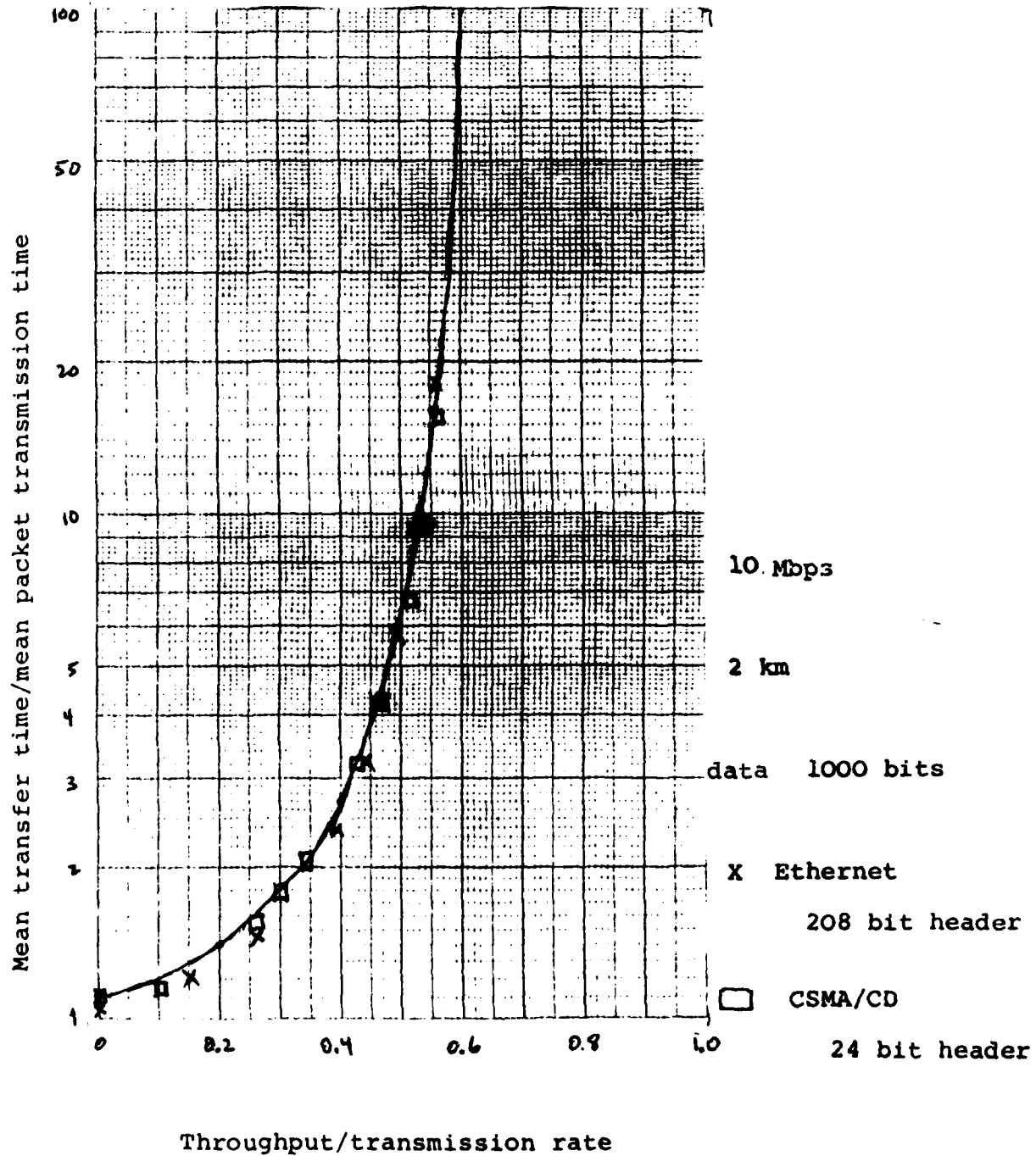


Figure 2.4 - THROUGHPUT/TIME DELAY CURVE FOR  $\tau = .088$

### 3. ASPECTS OF VOICE COMMUNICATIONS ON THE LOCAL AREA NETWORK, ETHERNET

#### 3.1 Summary

This Section describes simulation studies undertaken using the simulator described in the previous Section. The studies concentrate on voice communication over the Ethernet access protocol for local area networks. An introduction to packetised voice communication is presented which is followed by a presentation and justification of the delay-budget environment in which the results should be placed. These studies evaluate the maximum number of voice channels supportable by the communication system, and present the effects of system parameters on that maximum: packetisation time, speech activity detectors, and data compression.

The main results of these studies are the optimisation of packetisation time and a comparison between speech activity detection and data compression techniques.

#### 3.2 Description of Packetised Voice

Speech varies a great deal on the macro scale (seconds), from impulsive sounds, to simple tones, from virtual white noise to complete silence. But its structure starts to appear on a finer time scale [27]. Many phonemes will show a periodic wave

form in the order of several milliseconds, and an overall duration in the order of hundreds of milliseconds. The vast majority of the signal strength is confined to less than 4 kHz.

The simplest technique for digitising signals of this variety is Pulse Code Modulation. The signal is usually sampled every 125  $\mu$ sec (all frequency content above 4 kHz being removed to prevent aliasing) so that the full 4 kHz bandwidth is captured. The samples are then quantised, usually into 8 bits and usually through a non-linear scale (A law or  $\mu$  law) that ensures fidelity at low amplitude. The output of this encoder (codec), 8 bits every 125  $\mu$ sec, is transmitted serially at 64 kilo bits/second.

It is interesting to note that the smallest bandwidth analog channel over which this bit stream could be transmitted is 64 kHz, while the original signal could have been transmitted over a 4 kHz analogue channel [28]. The extra bandwidth is the cost of the digital signals greater resistance to noise.

The previous example points up the need for effective data compression. There exist a variety of schemes, of varying complexity and rates of compression [29], a discussion of which is beyond the scope of this report. In general, however, several observations are in order. Relatively inexpensive, and easy to implement schemes can reduce the bit rate to around 16 k bits/sec. More complex schemes for sending the frequency content of the signal or modelling the voice signal and sending

only its parameters can reduce the bit rate to the neighbourhood of 2 k bits/sec. The processing power needed to encode and decode the signal can be quite extensive, however, leading to the observation of a bit rate/processing trade-off.

A different technique for removing redundancy in spoken conversation is to stop bit transmission during silences, and transmit only during talkspurts by the use of speech activity detectors (SAD). There are two parameters to set: the signal detection amplitude below which transmission will stop, and the length of delay between detector level crossing and transmission cut off (the overhang). They must be balanced so as to remove as many bits as possible yet not clip off initial consonants, nor lose final unstressed portions of phonemes. In addition, the region of the codec's greatest sensitivity (around zero) will be partially removed, reducing fidelity. However, when using speech activity detection, a voice channel's average bit rate should decrease by approximately 60 per cent [30].

The simplicity of implementing a SAD system is slightly complicated by the need to clock the playback for those systems where delay might occur during transmission and confuse the length of silences. Also, not talking and the absolute silence of no playback sound different over a telephone, and one is used to the former.

In order to transmit this bit stream on Ethernet it is clear that

it must be divided into packets. What is not clear is the appropriate size of the packet, the time period for packetisation. Because the first bit of a packet can't be sent until the last is incorporated, there is an inherent delay involved in packetisation. Though Ethernet runs more efficiently for large packets, the bit stream will, unfortunately, be delayed longer.

In order to balance these two effects it is necessary to consider a total delay budget. There are three sources of delay on packetised voice transmission: packetisation, queueing for transmission, and transmission itself. It follows that large packets, system congestion, and a low transmission rate each lead to packet delays. But small packets lead to system congestion; so for a fixed delay budget, it may be possible to optimise the choice of packet size. Our simulation studies follow this line of reasoning.

Answering the question, "how much delay is too much?" for packetised voice communication, is difficult because of the inherent subjectivity of listening. However, surveys have been compiled on responses, to echo, noise and delay on telephone conversations [31], [32], from which a generalisation is clear. The effects of delay are much more significant in the presence of echo. While a one-way delay of 200 msec or so is acceptable when no echo is present, only 40 to 80 msec will start to be noticeable when echo is present. Added to this finding is the observation that standards in communication quality are rising.

The reader is invited to recall her reaction to conversation over a satellite link, where one-way delay is at least 250 msec and echo may be present.

Since the public telephone system exhibits echo, our simulation studies have been structured accordingly. Ethernet is a local access protocol, and a voice conversation extension to it would not be an improvement on existing techniques if it didn't link to the telephone network. Our studies assume echo may be present, even if Ethernet is not generating it.

Because Ethernet uses a multiplexed channel it cannot provide the delay characteristics of a dedicated channel, approximately 2 msec for a PBX. Our studies seek to show that acceptable voice quality can still be maintained on a much more efficiently used communications medium.

Some voice packets will arrive so late on Ethernet their data will be useless. This is a result of the random delay associated with queueing for use of the transmission channel. The effect on speech playback of packet loss is a packetisation-time long click (tens of milliseconds). The effect is slight, but will be worse for longer packetisation time. If one fixes a certain speech quality level for system design, it's the same as fixing an acceptable packet loss percentage - as long as packetisation time does not vary too much. Our studies treat packet loss as a measure of the quality of the transmission system, and fix an



acceptable level for the system.

### 3.3 Formulation of Constraints for Simulation Studies

We assume an Ethernet environment for our packetised voice system which included a server that connects to the public telephone system. The result is a much tighter limit on one-way delay (D) within the Ethernet. This delay is assumed fixed by system performance criteria. Because not all packets will, or need to, achieve this delay limit, a loss percentage (L) is accepted. This percentage is also assumed fixed by system performance criteria.

Studies are conducted in each of three cases: no data compression, data compression, and speech activity detectors. Identical procedures are followed and the merits of the three cases compared.

What we attempt to maximise is the number of active voice channels that Ethernet can support, that is, the number of simultaneous conversations. A voice channel is a communication line for one speaker; two together constitute a conversation. We assume independent talkspurt statistics for the two channels, which leads to more simultaneous talkspurts than would occur in a real conversation, and pessimistic predictions of the number of simultaneous conversations supportable.

In summary, let  $p$  be the packetisation delay,  $t$  the transmission delay, and  $q$  the queueing delay for a particular packet. If the packet is lost due to repeated collisions let  $q$  be infinite.

Then the speech quality constraint can be expressed as

$$(1) \text{ Prob } \{ p + t + q < D \} < 1 - L$$

where  $D$  and  $L$  are fixed, and the probability function uses all packets as its domain. We find the largest number of simultaneous voice channels  $M$ , which the system can support without violating (1). We then vary the packetisation time,  $p$ , to find the largest  $M$  among the  $M$ 's. This optimisation of packetisation time is repeated for each of the three cases mentioned above.

The choice of values for  $D$  and  $L$  must be made. For  $D$  we pick the most conservative bound on voice quality consistent with [31], 40 milliseconds, which follows the work of [33]. It is a compromise between allowing for echo and delays in the telephone network and providing a loose enough bound for flexible Ethernet system design. For  $L$  we pick a conservative bound of  $\frac{1}{2}$  per cent, bearing in mind that it must reflect acceptable quality when only talkspurts are transmitted, as with SAD systems. An evaluation of the sensitivity of our results to this value is presented.

Other authors have considered the problem of packetised voice on CSMA/CD systems, both through simulation [34], [35], and experiment [33], [36]. There is also a helpful overview of the problem [37].

### 3.4 Results of Simulations

In Figure 3.1 are shown the results of the simulations involving 64 k bit/sec voice channels. The upper curve represents a point of comparison. It shows the maximum number of channels supportable by the system if there were perfect scheduling and no queueing delay. The curve increases monotonously with packetisation time, essentially reflecting the ratio of useful data to header as packet size increases. Because the packets are larger, less of the communication medium's time is spent transmitting header, and more voice channels can co-exist. Because queueing delay is defined to be non-existent, and transmission time is negligible, the voice quality constraint (1), is not violated till packetisation time equals 40 msec.

The lower curve shows the results of simulation: the maximum number of voice channels that can be supported by Ethernet without violating the voice quality constraint, as a function of packetisation time. Several observations are in order. The values are about 60 per cent of the theoretical maximum, due to contention. This is expected because in these parameter ranges, the average packet delay should increase very rapidly once beyond an Ethernet utilisation of 70 to 85 per cent of maximum. Since high packet delay will violate the quality constraint, its effect is clear. The parameter of interest is the alpha value of Section 2, which varies with packetisation time, but is about .05 at 10 msec and about .02 at 30 msec.

Note that the curve indicates a maximum  $M$  of eighty channels, for an optimum packetisation time of 25 msec. The fact that an optimum exists is of interest, and can be understood as the result of two effects. Small packetisation time means more packets and channel attempts, hence more queueing delay due to contention. But large packetisation time puts a tighter bound on allowable queueing delay. The existence of this optimum follows in relatively straightforward fashion from the voice quality constraint (1), and Ethernet's greater efficiency with longer packets.

In Figure 3.2 are shown the results of the simulations involving 24 kbit/sec voice channels. The same general behaviour as Figure 3.1 is seen here. The upper reference curve with no queueing delay is monotonously increasing. The lower simulation curve shows a maximum of 155 voice channels at an optimum packetisation of 25 milliseconds.

In Figure 3.3 are shown the results of the simulations involving 64 kbit/sec voice channels equipped with speech activity detectors. Following [30], the mean talkspurt length is set to 1.3 seconds and the mean silence length to 1.7 seconds. As mentioned in Section 2, the talkspurts and silences are modelled as exponentially distributed. The same general behaviour as in Figure 3.1 and Figure 3.2 is seen here. The simulation curves show a maximum of 175 voice channels at an optimum packetisation time of 25 milliseconds.

In Figure 3.4 is shown a comparison of the simulation curves from Figures 3.1, 3.2 and 3.3. They all achieve their maximum for the same value of packetisation time, showing that the optimum value is robust in the face of system modifications. Also, these modifications, data compression and SADs, can more than double the number of simultaneous voice channels supportable by the system. The design lesson learned here is that some modification of the original 64 k bit/sec voice channels is highly desirable. But more may be concluded. The data reduction from 64 kilo bits/second to 24 kilo bits/second is by a factor of 3. In a naive way, one might expect a corresponding increase in the number of voice channels supportable to be by a factor of 3. In fact, the increase is not quite by 2. Actually this comes as no surprise since data reduction does not affect the number of packets that have to be transmitted, nor do packet headers diminish in size. However with speech activity detectors, the increase to be expected is 2.3 (the ratio of the talk plus silence periods to the talk period). The actual increase is almost exactly that.

Because SAD techniques reduce the number of packets sent, they blend particularly well with Ethernet, where each new packet raises the probability of collision for the others. As a result, more channels can be accommodated by SAD techniques than by a moderate rate of data compression. Given the simplicity of SAD techniques and the expense of more extreme rates of data compression, the design lesson learned here is that initial

concentration should be on the former.

In Figure 3.5 is shown an investigation of the sensitivity of our results to variation in  $L$ . Choosing the optimum point for 64 k bit/sec voice channels, 25 msec packetisation, the number of voice channels is varied. The percentage of packets that achieve a total time delay of  $D = 40$  msec is plotted. One can see that  $L = .5\%$  implies a limit of 89 voice channels. Adding more voice channels involves a rapid degradation of voice quality, beyond the point of any use. Given the steepness of the descent, it seems wise to design with conservative limits, and avoid this region of the performance curve.

Figure 3.1 Optimum packetisation (64 kbit/sec)

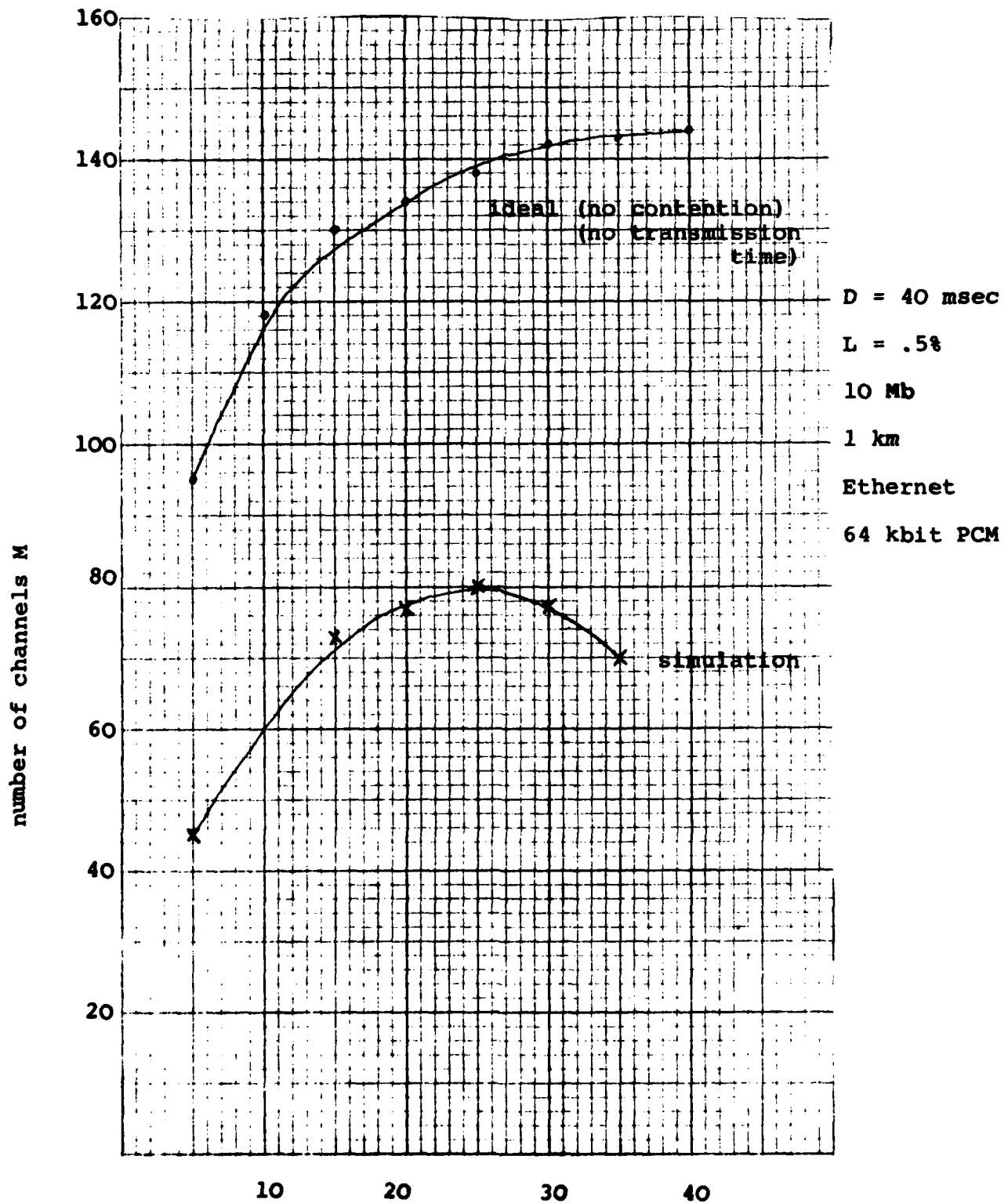


Figure 3.2 Optimum packetisation (24 kbit/sec)

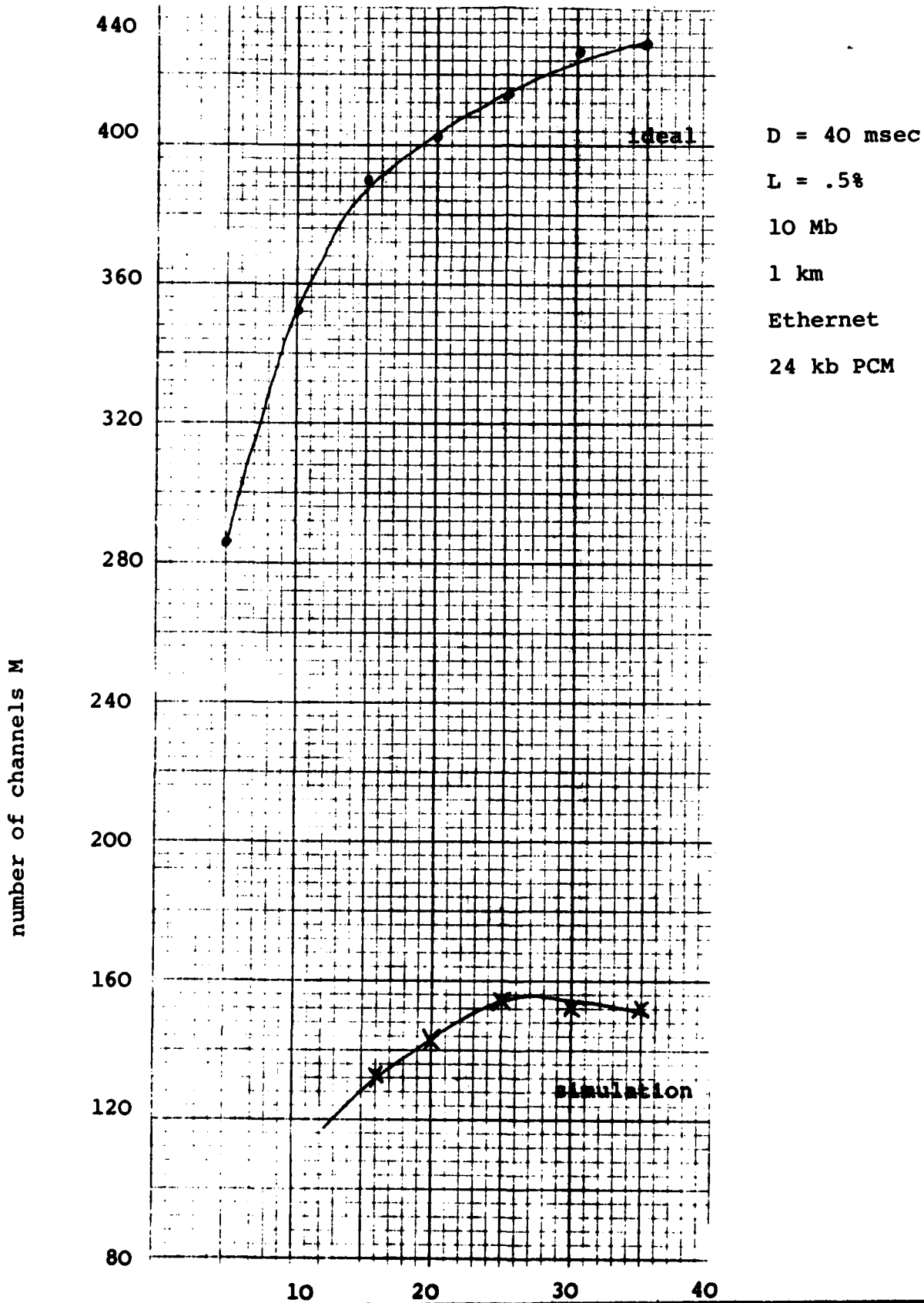




Figure 3.3 Optimum packetisation (SAD)

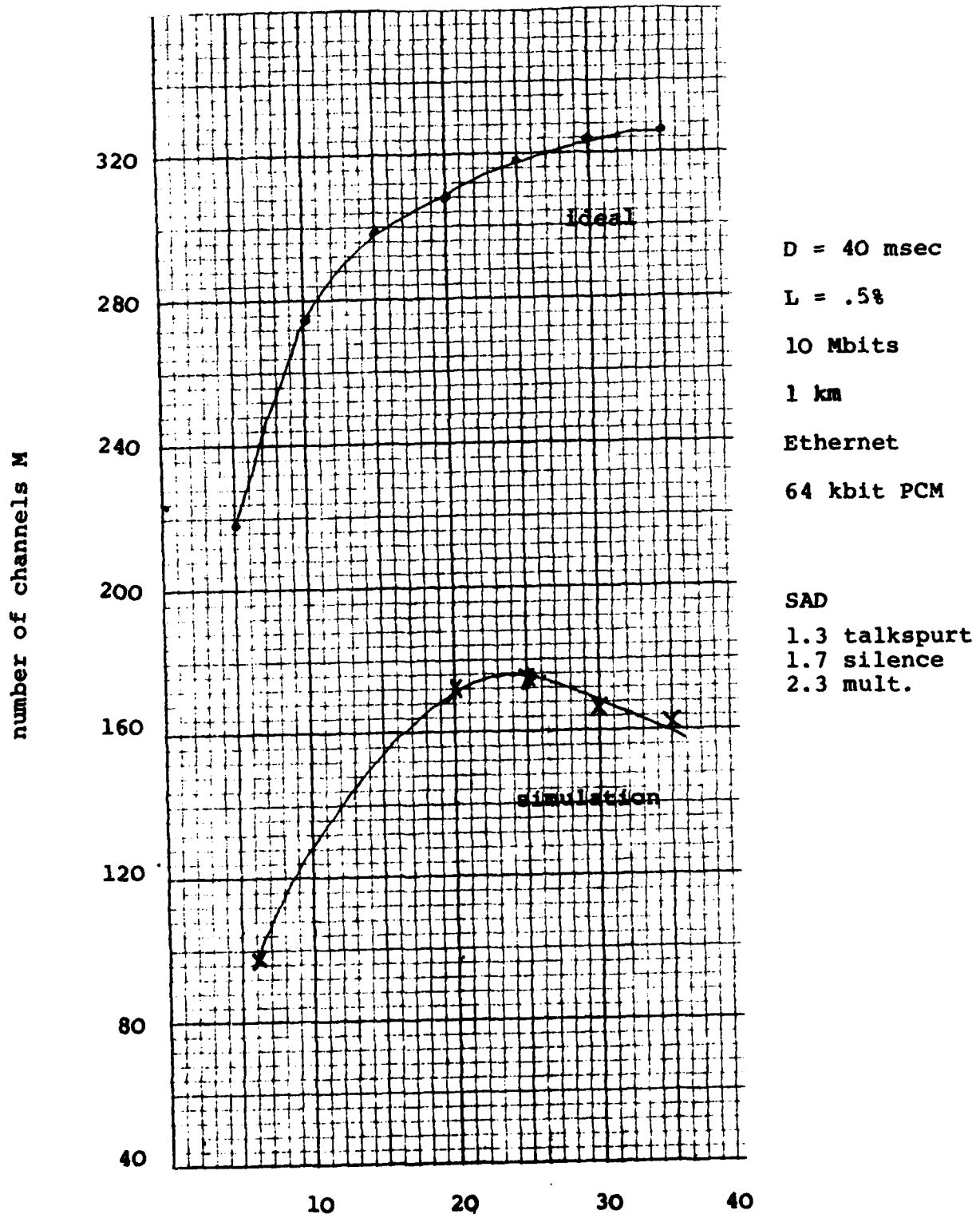


Figure 3.4 Comparison of optimums

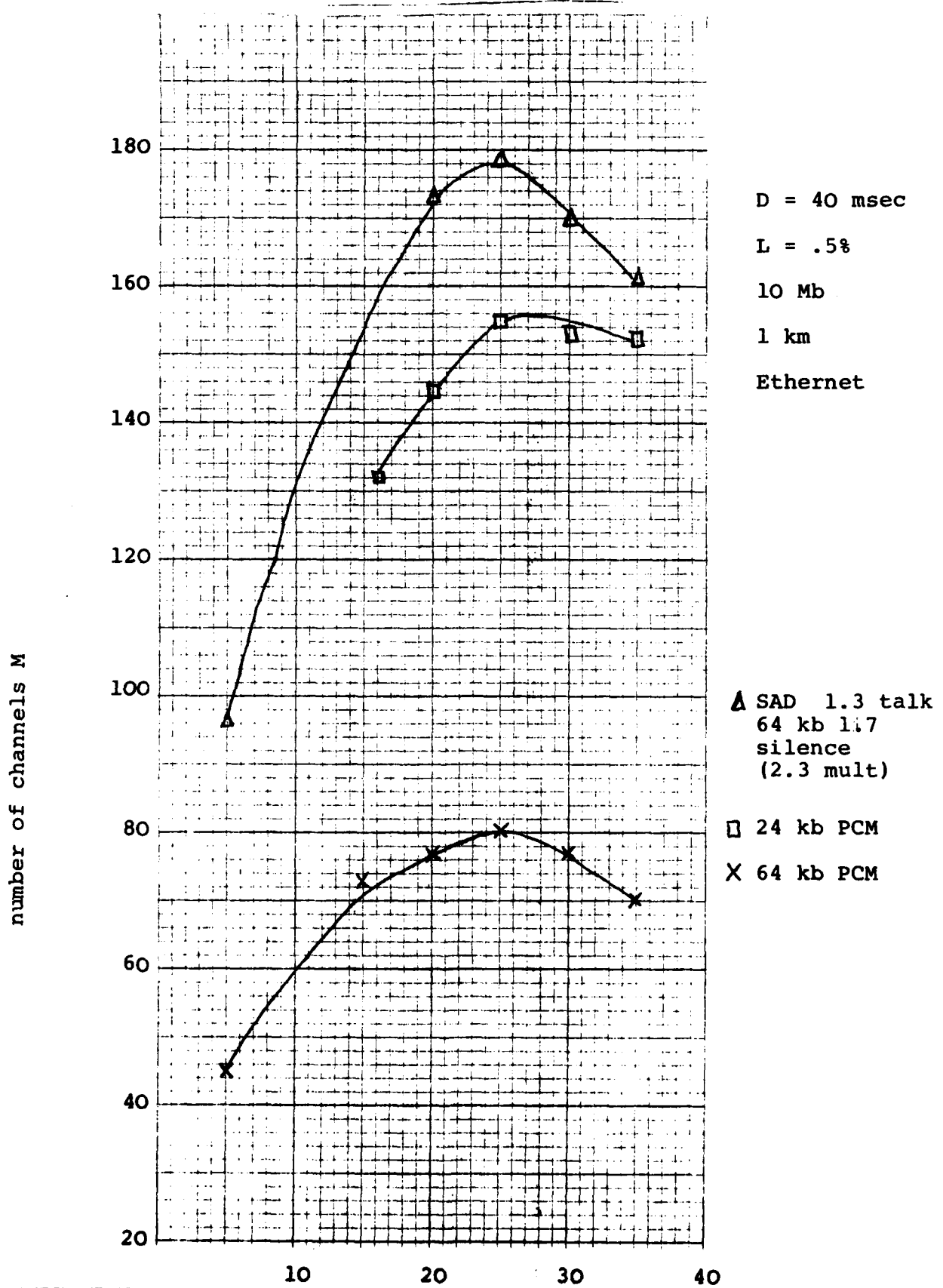
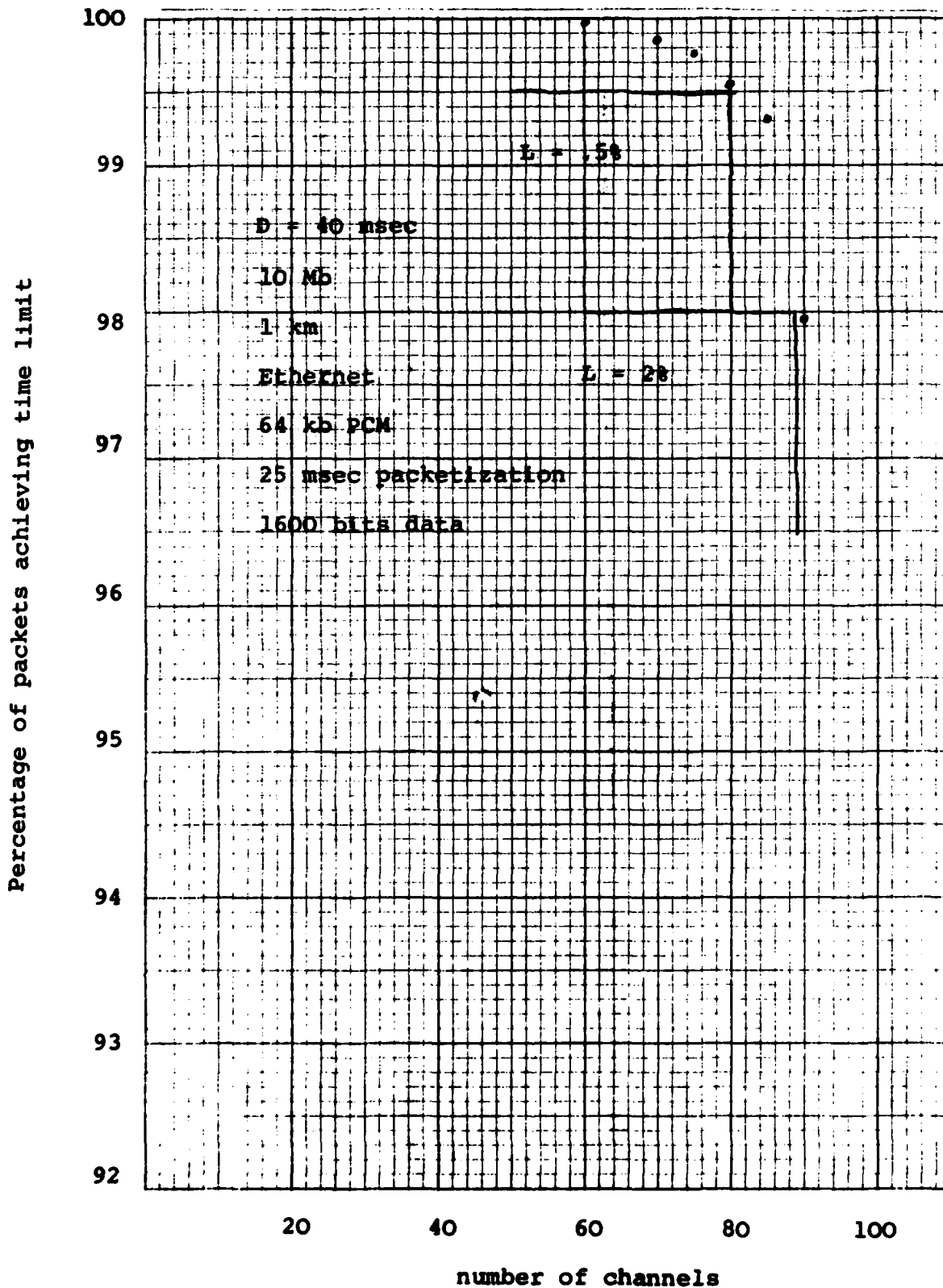


Figure 3.5 Sensitivity of results to L



#### 4. MODELLING AND SIMULATION OF A TOKEN PASSING RING (BUS)

##### 4.1 Summary

This Section summarises the work done at the Polytechnic of Central London to model and simulate a token passing ring based on the National Physical Laboratory multicomputer ring [23]. This is a high-speed, token-passing, byte-parallel communications structure for interconnecting micro or mini-computers in a closely-coupled configuration. The model developed, however, is general enough to simulate other token passing ring or bus architectures. To do the modelling work, the graphics aided interactive distributed computer modelling and simulation package MICROSS has been extended. The MICROSS communication scheme is based on the store-forward principle. In order to simulate the token-passing ring, appropriate extension to the existing package has been coded and debugged. The results presented in this report made use of the extended facilities.

#### 4.2 Token Passing Ring Structure

The token passing ring under consideration consists of up to 255 nodes connected by means of a data ring to produce a synchronous communication system.

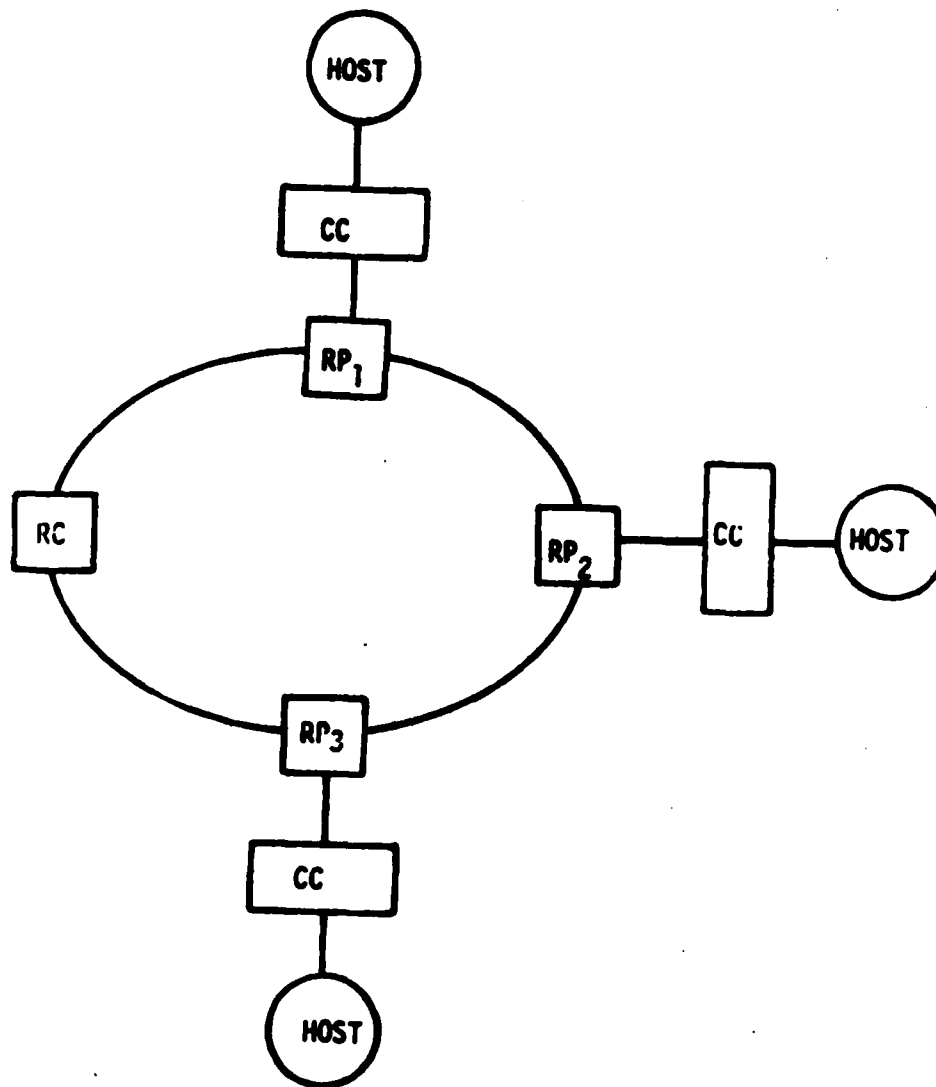
Data is transferred as a series of packets moving around the ring. The structure is shown in Figure 4.1.

Three basic hardware modules are involved:

- (1) a Ring Closer (RC)
- (2) a Ring Port (RP), and
- (3) a Channel Controller (CC)

Essentially the RC generates the system clock, handles initialisation, and electrically "closes" the ring. An RP and CC combined comprises each node in the ring. The RP acts both as a relay station and as a send/receive unit, with the CC acting as a parallel interface between the RP and its associated host computer.

The entire ring hardware is physically localised, being confined to a single location in a building. RP's can be grouped into clusters with short (typically 2 metres) ring cables interconnecting the clusters. Each CC is housed with its associated



RC- Ring Closer  
RP- Ring Port.  
CC- Channel Controller

Figure 4.1

computer and links to the ring by means of a long (typically 5 metres) multiway cable.

Here we list the main features of the token passing ring modelled.

- \* The ring may have up to 255 nodes
- \* Data for transmission on the ring is assembled into packets. These can be of any length, up to 65 bytes. Control overhead is a constant 5 bytes.
- \* RPs can act as both sources and destinations.
- \* Ring access at each RP (node) is governed by a single, circulating token contained in a packet header. This token is placed onto the ring by the RC at ring initialisation time. The claiming of this token by any RP gives that RP the right to place a packet onto the ring. When the RP has finished doing so, it must pass the token downstream.
- \* A reserved feature in the design guarantees access to an RP acting as a destination within a known maximum time. This is important where many RPs are trying to send to one particular destination. A hardware queueing mechanism ensures that a would-be source has a finite access time to any selected destination.
- \* A packet on the ring is delayed by one clock period at each Ring Port (RP).

- \* Consecutive packets on the ring will have no gap between them (i.e. the header of one immediately follows the terminator of the other packet).
- \* In each RP, data is assembled and deposited into 64-byte output and input buffers (one of each).
- \* Error detection is incorporated.
- \* Overall, the CCs act as interfaces between computers and the ring. Each computer uses a slave processor - part of its own system - to handle transfers across this interface. Data is read from each RP by polling action.

#### 4.3 The Model

The ring is considered to consist of a number of nodes linked together in a loop with packets being generated and sent between nodes as complete units. Each node is envisaged as two inter-connected processors as shown in Figure 4.2.

- (1) Host Processor (Host) which combines the functions of the Host and Channel Controller in the real system, and
- (2) Communication Processor (Comm) which mimics the functioning of the Ring Port.

The Ring Closer module appears in the simulation model as an extra delay between the first and last nodes.



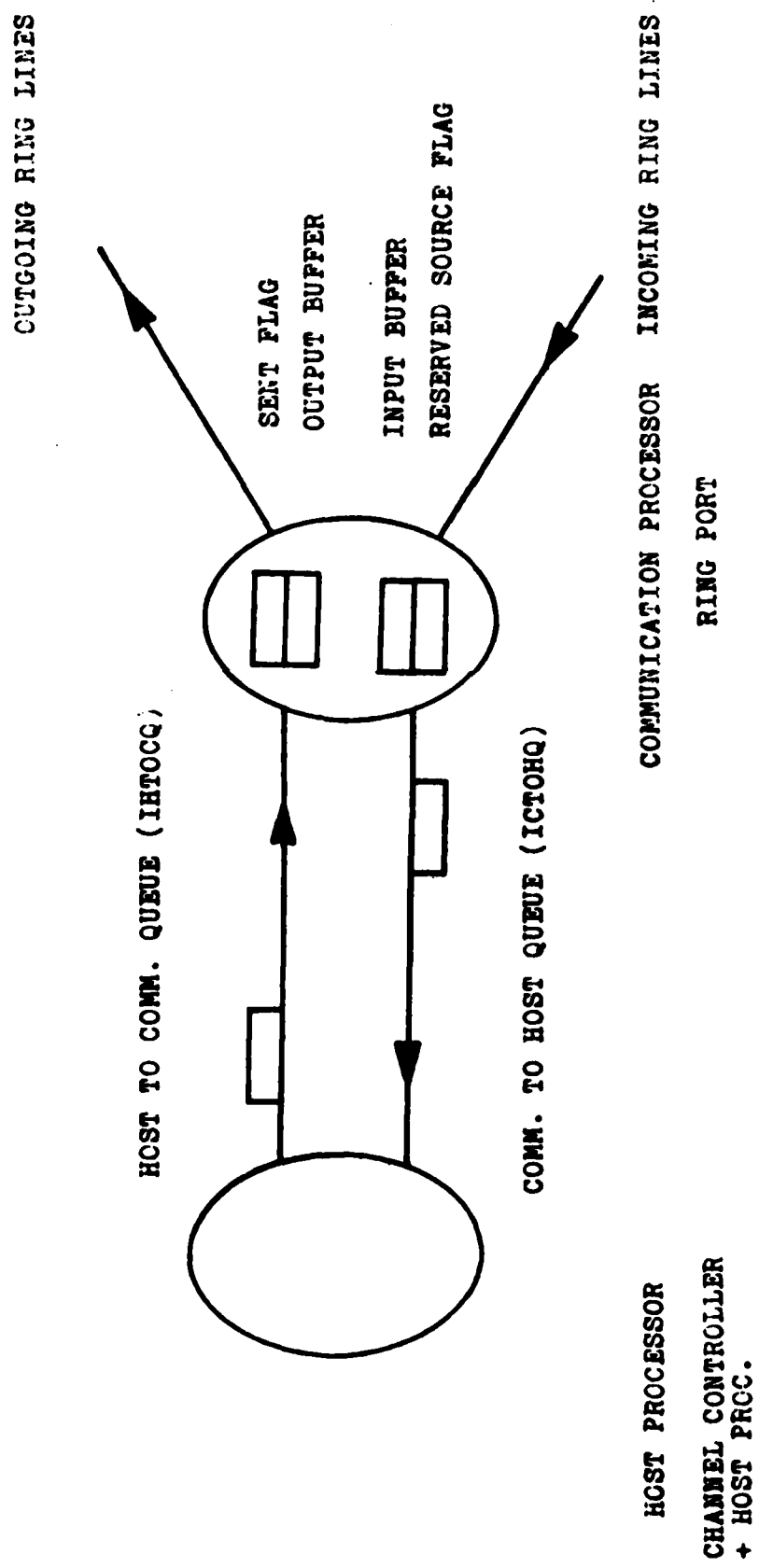
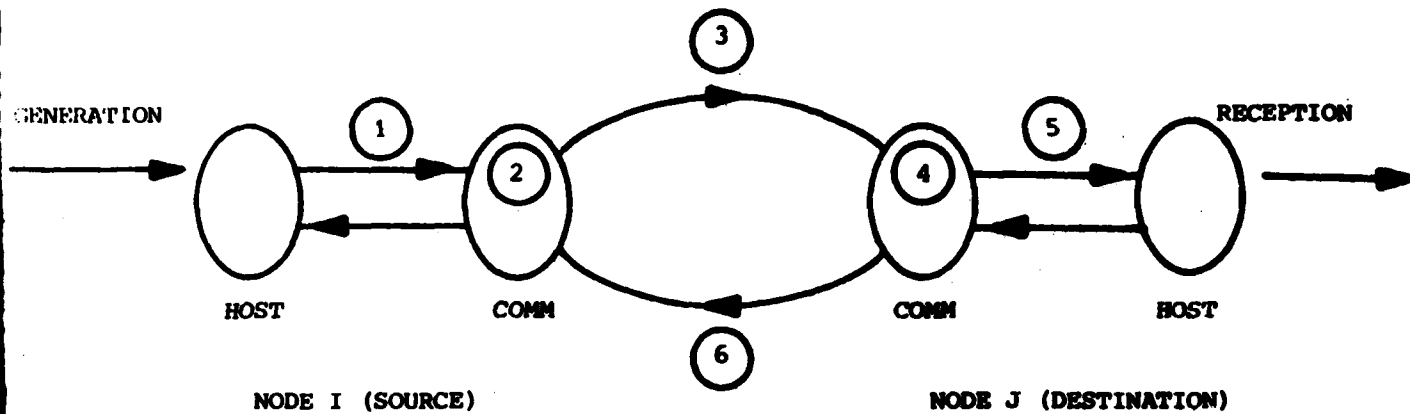


Figure 4.2 Node Structure

(X) PACKET STATE X



STATE	DESCRIPTION	IDTX	IS	ID	IC	IN
1	Newly generated, in queue to comm.proc.	M	I	J	I	O
2	In output buffer waiting to be sent	M	I	J	I	I
3	On ring (outgoing trip)	M	I	J	I	J
4	In input buffer at destination	M	I	J	J	I
5	In queue to host proc. at destination	M	I	J	J	O
6	On ring (return trip)	-M	I	J	J	I

IDTX = packet ID  
 IS = source ID  
 ID = destination ID  
 IC = current node ID  
 IN = next node ID

FROM	TO	DELAY	
generation	stage 1	L/host speed (*)	(L = length of packet in bytes)
stage 1	stage 2	L/line speed	
stage 2	stage 3	1 clock period	
source	destination	(J-I+L) clock periods	
stage 3	stage 4	1 clock period	
stage 4	stage 5	L/line speed	
stage 5	reception	L/host speed (*)	
destination	source	(I-J) clock periods	

\* Not implemented in this version

Figure 4.3 State Diagram of a Packet

During the simulation, packet generation and its passage through the system is controlled by various delays and the status of the relevant flags. The token is represented by a packet whose identification is zero and whose length is one byte. A diagram outlining the states of a packet is given in Figure 4.3. This also has expressions giving minimum delays for the various transitions.

The operation of the package takes place in three main phases:

- (1) definition
- (2) simulation, and
- (3) reporting.

#### Definition

During this phase the user is requested to enter the characteristics of the ring by specifying the following variables:

Clock frequency	CLOCK F	MHz
Number of nodes	NODES	
Host processor speed	HSPEED	k inst/sec
Host to comm. line speed	HTOCS	k bytes/sec
Host to comm. line length	INTOCQ	packets
Comm. to host line speed	CTONS	k bytes/sec
Comm. to host queue length	ICTONQ	packets
Ring closer module delay	IRCND	clock period

Packet length	MLTH(I)	bytes
Packet length distributor	MLDT(I)	
Packet inter arrival time	PIAT(I)	seconds
Inter arrival time distributor	IATDST(I)	
Probability of destination J		
for source I		

#### Simulation

On entering the simulation phase the user is asked for an initial simulation period during which simulation may occur. A flow chart of the simulation phase is given on Figure 4.4.

#### Reporting

On simulation termination, a report is generated giving the main features of performance:

- Average packet response time
- Packets generated at a node
- Packets received at a node
- Packets delivered from each node
- Occurrence of reserved flag sets.

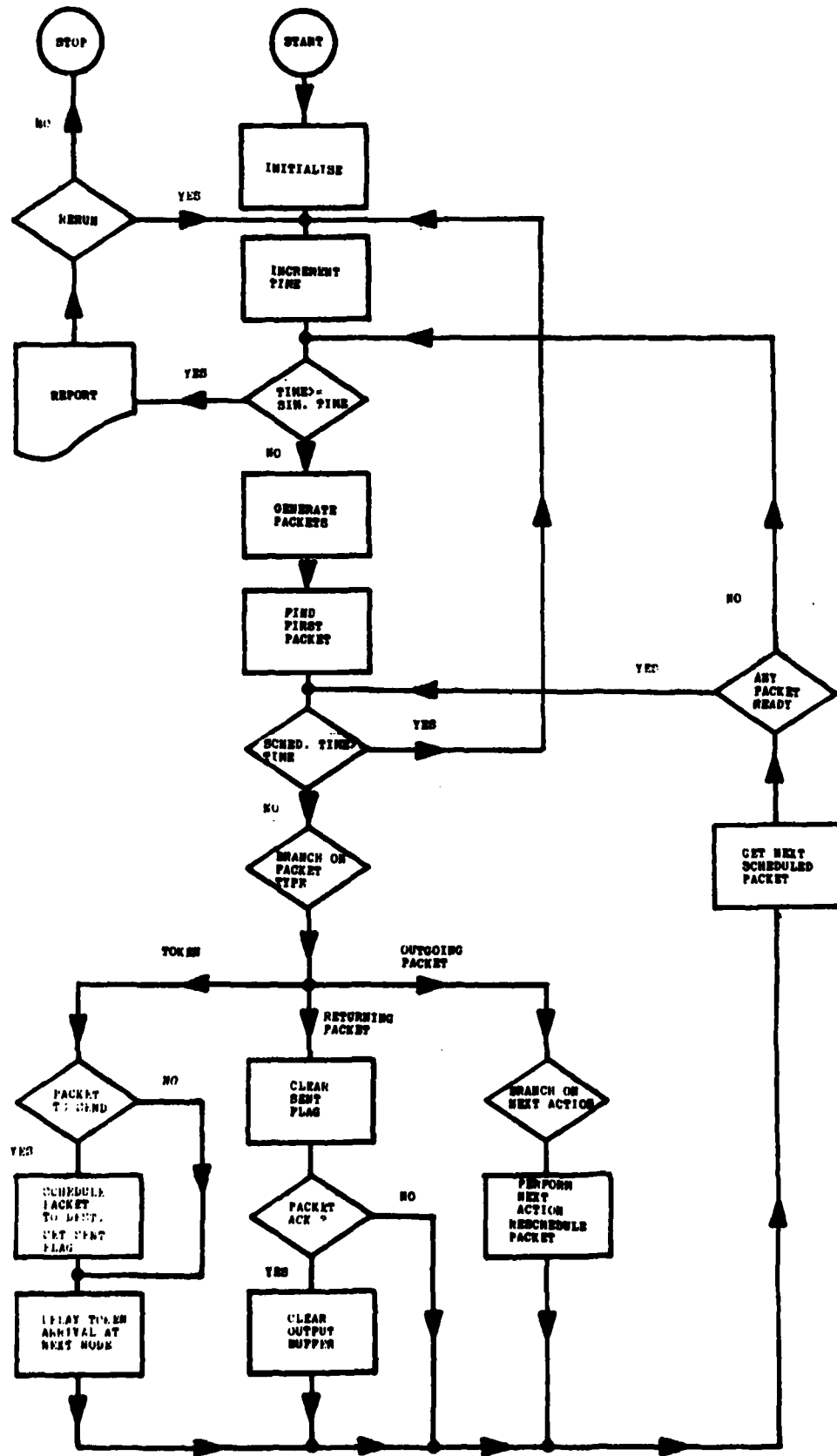


Figure 4.4 Simulation Flowchart

#### 4.4 Token Passing Ring Operation Modelling

##### Handling External Packets

The transaction is treated as an external packet only if the CC (Host) is not busy with another task and also there is enough storage space (queueing space). Once an external packet is accepted at the host it can be directly transferred to the output buffer (OB), if OB is free and if the packet is the only one at the host. Then it is deleted from the host storage. Otherwise the packet waits in a queue.

##### Handling the External Packet Arrival

- (1) A packet attempts to enter the ring
- (2) if CC = free then
- (3) handle the packet
- (4) add the packet to the output queue (OQ)
- (5) if output buffer (OB) = free then
- (6) transmit to OB
- (7) delete the packet from OQ
- (8) else
- (9) else
- (10) return

##### Arrival of the Token at a Ring Port (RP)

If the OB at an RP is empty during the detection period of a token arrival, then a new token is generated to be scheduled

to arrive at the next RP. Otherwise, first the packet in the output buffer is scheduled to arrive at the destination, then the token is generated and scheduled to arrive at the next node.

The important point is the simulation of time on the ring. For example, the time difference between the detection of a token and its departure is one clock period, in the case of an empty OB. Otherwise, the token is delayed by

$$t = (L + 1) * C \text{ microseconds} \quad 4.1$$

Here L is the number of bytes in the packet and C is the clock period in microseconds. The arrival of the header byte at its destination, on the other hand, is delayed by

$$t = (m + 1) * C \text{ microseconds} \quad 4.2$$

where m is the number of RP's between the source RP and the destination RP, inclusive.

Note that the packet in an OB is normally inactive and activated only by the arrival of the token, whereas a packet at the host or a packet in an IB is active, waiting for the CC to be free.

#### Message transmission routine

- (1) token arrives
- (2) if OB is full then transmit packet
- (3) else

- (4) transmit the token (schedule)
- (5) return (update the status of RP)

#### Receipt of Packet at RP

The packets put on the ring at the source RP's are received at the destination RP's. The receive operation can take place only if the IB is free. The following procedure simulates the receive operation. It includes the "reserved" feature where the destination is reserved for the first source requesting a receive operation but finding the IB in a busy state.

#### Receive procedure

- (1) packet arrives
- (2) if IB busy then
- (3) if destination RP not reserved then
- (4) set destination RP reserved for source RP
- (5) else (7)
- (6) else receive message (schedule)
- (7) send acknowledgement (negative or positive)  
to source node (schedule to complete round-trip for the source)
- (8) return

The receive operation at (6) is simulated in a time given by Equation 4.1 which is the time that the next header (or token) arrives at the RP.



### Arrival of Round-Trip Acknowledgement

The detection of an Acknowledgement indicates either positive, causing the OB to be free for the next packet, or negative, requiring re-transmission during the next arrival of a token.

#### Acknowledgement handling routine

- (1) packet returns
- (2) remove packet from ring
- (3) if Ack = positive then
- (4) set OB to free
- (5) return

### Receipt of a Message at the Host

The packet received at the IB of RP is accepted at the CC, therefore at the host, if the host is free. Upon reception at the host the IB is freed. Another receive cannot take place at an RP as long as the IB is busy.

#### Host receive procedure

- (1) IB is busy (message available request)
- (2) if CC (or host) = free then
- (3) transmit the message across the RP-CC  
interface
- (4) set IB to free
- (5) return

Scheduling is not required since this is the ultimate destination for the message.

### Token Handling

The token is considered to be a special packet whose identification number is 0 and length is one byte. On passing a node the token is delayed by either one clock period or by a period proportional to packet length if a packet is to be transmitted.

## 4.5 A Typical Simulation Run

### 4.5.1 Experiment Constants

Here a typical simulation run is presented where the experiment is carried out with the following constants:

- |                                       |   |
|---------------------------------------|---|
| a) ring speed:                        | 8.33 Mb/s                                 |
| b) queue length:                      | 1 packet                                  |
| c) line speed:                        | 100-1000-10000 Kbyte/s                    |
| d) Ring Closer module delay:          | 10 clock periods                          |
| e) the distribution of packet length: | fixed                                     |
| f) the distribution of destination:   | equally likely                            |
| g) the distribution of packet         |   |
| Inter Arrival time:                   | Poisson with average<br>100 $\mu$ seconds |

#### 4.5.2 Definitions

The simulation runs allow us to determine the following figures of merit.

##### Response Time

The response time refers to elapsed time between the packet generated at the sender and accepted by the destination.

##### Throughput

Throughput is a measure of the packet carrying capacity of the system. This can be measured in absolute terms, as the number of packets delivered by the network per second. Throughput reaches a maximum dictated by the ring capacity.

##### Efficiency

In order to better interpret the absolute throughput values given above, two definitions are introduced:

- (1) Efficiency E and
- (2) Throughput Efficiency T.

The Efficiency E is defined as the percentage ratio of throughput to maximum ring capacity.

$$E (\%) = \frac{\text{Throughput (Pkts/sec)}}{\text{Max. Ring Capacity (Pkts/sec)}} \times 100$$

The second definition, Throughput Efficiency T refers to what percentage of packets that are introduced at all the nodes eventually gets delivered.

$$T = \frac{\text{Throughput (Pkts/sec)}}{\text{Max. Packets generated (Pkts/sec)}} \times 100$$

#### 4.5.3 Performance Experiment Figures

In Figure 4.5 the simulation results for IAT = 100  $\mu$ second are shown. Figure 4.5a plots the variation of response time as a function of ring size for three line speeds indicated. For slow line speeds, there is not much change in response time for larger networks. For faster line speeds, one observes a nearly linear dependence, since the ring speed becomes more influential.

Efficiency Figure 4.5b shows that E is linearly dependent on size for slow line speeds. Otherwise E grows rather fast to approach maximum efficiency for faster line speeds. For small ring sizes there is not enough traffic generated to fill the ring, hence efficiency is low. Once the ring becomes saturated, however, adding more nodes does not improve the situation. Figure 4.5c plots the throughput efficiency T.

It can be observed that by increasing the line speed the traffic is pushed onto the ring which becomes the limiting factor. Figure 4.5d plots efficiency (E) divided by response time as a

function of number of nodes. For faster line speeds, it is interesting to note the peaking in Figure 4.5d. In reality the peaking is more prominent than it appears on logarithmic scale. This happens near the optimum ring size given in [23].

+ = LINE SPEED 100 KBYTES/SEC

□ = LINE SPEED 1000 KBYTES/SEC

○ = LINE SPEED 10000 KBYTES/SEC

I.A.T. = 100 MICROSECS

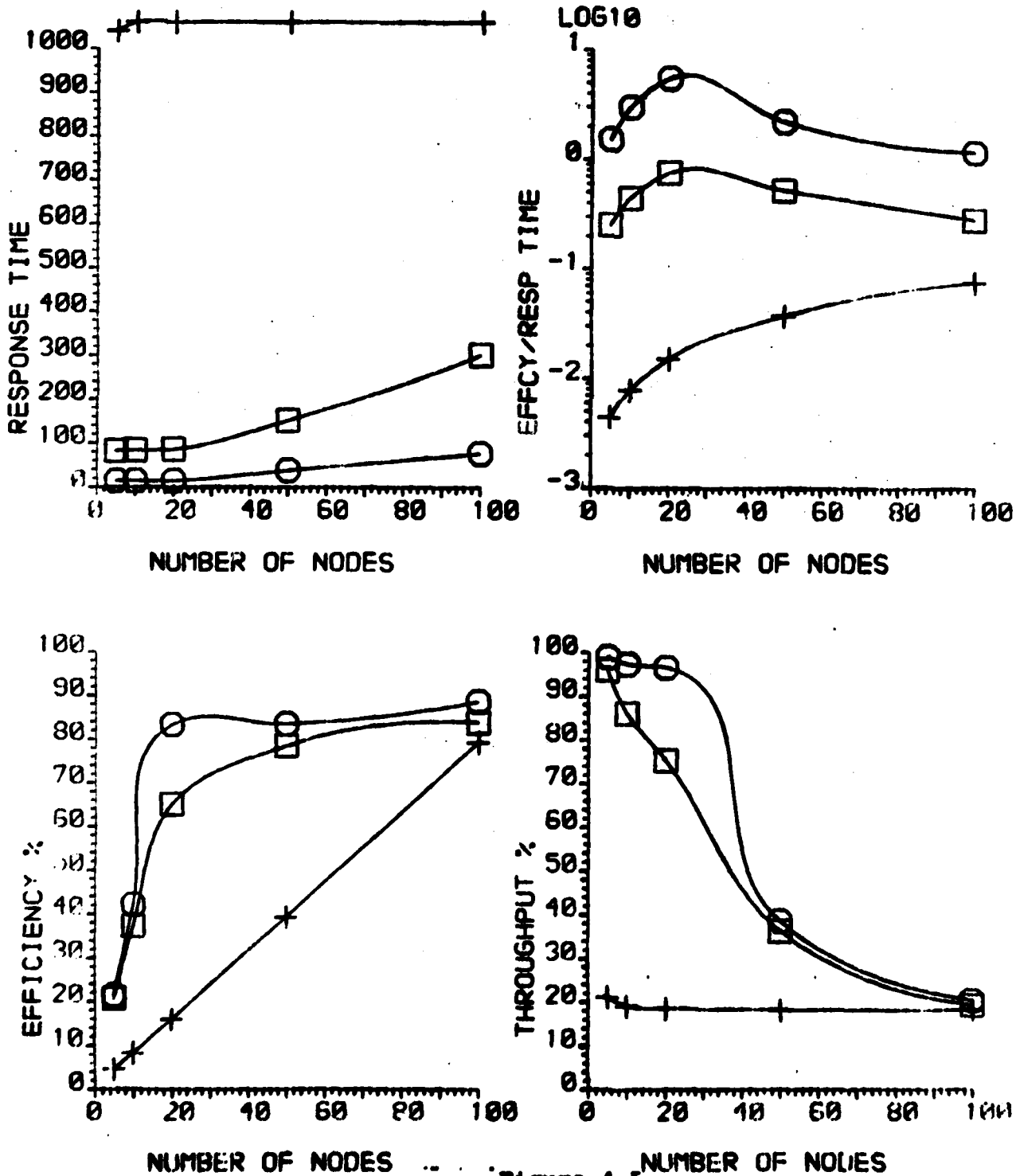


Figure 4.5

## 5. MODELLING AND SIMULATION OF A CIRCULATING SLOT RING

### 5.1 A Slotted Ring Local Area Network

This Section gives a general description of a Slotted Ring LAN, namely the Cambridge Ring82 implementation as defined by the Joint Network Team [38].

#### 5.1.1 Principles

At the lowest level the communication link comprises a closed ring of cable and active repeaters. The delay in signal propagation through the cable and repeaters means that the system may be considered as a continuously circulating shift register. Each 100 metres of cable causes a delay of 450 nsec and so may be thought of as storage of 4.5 bits for a 10 Mhz signalling frequency.

The circulating storage contains a fixed number of bits for any particular configuration of the network; in order to provide communication a regular structure is imposed on these bits by a Monitor. This consists of a small number of slots, all the same length, travelling nose to tail, with a single gap of a few bits to mark a complete cycle. The start of each slot is indicated by a leader bit which is always one, and the gap contains only zero bits. Very simple algorithms may be used to synchronise this structure.

The signalling frequency is nominally 10 MHz but may vary slightly to produce an integral number of bits in circulation. To allow for adjustment of the amount of circulating storage a variable size shift register is used to provide additional 'padding'. The variable padding is set to permit an integral number of slots and a small gap.

The repeaters regenerate the signals on to each section of the ring, allowing rings up to several kilometres to be constructed, and providing for different cable types to suit the environment of each part of the network. The repeaters are also the points of attachment to the network for communicating devices. They demodulate signals from the ring and present them to a station which may alter them before they are modulated on to the ring again. Because the signal is remodulated at every repeater future provision of fibre optic components is a natural enhancement.

An active system of this kind lends itself to the implementation of a number of low level error detection and correction techniques that can be used to quickly localise faulty devices.

Since the operation of the repeaters is essential to ring integrity they are powered along the ring cable and are thus independent of the station and attached device.

#### 5.1.2 Access

Communication takes place between stations which synchronise



to the slot structure. The unit of transmitted data between stations is known as a minipacket and occupies exactly one slot. Each minipacket is individually addressed, carrying eight bit source and destination addresses. Sixteen bits of user data are carried, and in addition two 'type' bits that provide for flexible user identification or framing of the data. The first bit following the slot leader bit is a full/empty marker, used to control access to the slots. Following the user data and type bits are two response bits.

A station wishing to transmit waits until an empty slot arrives; it then marks it as full, inserts the addresses and data, and initialises the response bits. The transmitter is only allowed one minipacket in flight at a time and it counts passing slots to determine when the minipacket it transmitted has returned. The transmitter marks the returning slot empty and copies the response bits. By requiring that the returning slot be emptied round robin scheduling is introduced in which, however heavily loaded the ring, each slot passes around the stations giving them an opportunity to transmit within a determined time. The response bits are used to carry back low level flow control information to the transmitter. The receiver may be physically or logically absent from the ring and in this case the transmitter will see that the minipacket has been ignored. The receiver may also indicate that although he is present he is not listening; a 'source selector' may be set to 'anyone', 'no-one' or 'n'. When it is set to no-one or a value of n different to that of

the transmitter address the response bits indicate 'not selected' to the transmitter. Thus the receiver is able to listen to all transmitters (anyone) and multiplex the incoming minipackets individually, or may concentrate on one transmitter ( $n$  = transmitter address), allowing very simple implementation of block protocols. The receiver can also generate a 'busy' response when unable to process the minipackets as fast as the transmitter tries to send them. A transmitter seeing the busy response sends the data again in a new slot as often as required. A minipacket that has been successfully received is marked as 'accepted'.

Some refinements to this basic mechanism enhance its performance and reliability.

Whenever a transmitter receives a response other than accepted it is not permitted to transmit immediately but must wait for the ring structure to cycle around. Second and subsequent unsuccessful transmission attempts cause the transmitter to be backed off for 15 ring cycles. This prevents the ring being swamped with useless traffic. Further, the delay is made traffic dependent by requiring that ring cycles are only counted for backoff purposes if the gap is immediately preceded by an empty slot. The round robin scheduling puts an upper limit on this delay while the variable backoff produces a system in which efficiency improves under load.

When a minipacket returns a transmitter is able to check for consistency with the minipacket that was sent out. If a discrepancy is found then the response seen by the attached device is 'transmit error', overriding the response within the minipacket.

A repeater and station together make up a Node that is common to all devices communicating on the network. Logic is required to interface the Node to any particular device and this logic is called an Interface Unit [38].

#### 5.1.3 Maintenance

In addition to the error detection used by transmitter facilities are included to continually monitor the entire system.

Every slot includes a parity bit that is checked and maintained by all stations. A station that detects a slot with faulty parity corrects it and also sends a fault message in the next empty slot to destination zero. The fault message contains the address of the sending station and so indicates the section of the ring where the fault occurred.

The minipacket structure includes a bit, the Monitor Pass bit that is set by a transmitter when it fills a slot. This bit is always cleared by the Monitor on passing slots. If the Monitor detects a slot that has this bit cleared but is still

marked full then it marks the slot empty. It is therefore impossible for a fault to cause a slot to become permanently full.

The Monitor is able to detect errors that interfere with the permanent slot structure and rapidly reinstate the correct structure in a non-destructive way. Large numbers of errors, perhaps caused by a power dip, cause the Monitor to reinitialise the network, resetting its basic frequency.

A Monitor can also fill empty slots with random address and data as they pass and check them as they return if they are still marked empty. In this way the Monitor keeps the performance of the ring under continuous surveillance and can give warning of incipient faults.

#### 5.1.4 Performance

The access control rules require that a transmitter has only one minipacket in flight at a time, and also that a slot is emptied after it is used. This specification also requires that the slot after the one emptied be allowed to pass. Thus the maximum slot utilisation that a transmitter can achieve is one in every  $(n + 2)$  slots. Since there are 16 data bits in every 40 bit slot this determines the maximum point to point data transmission rate to be  $4/(n + 2)$  Mbps. For a one slot ring the value is 1.3 Mbps. The round robin scheduling

means that this rate can be achieved simultaneously by two pairs of communicating Nodes. Increasing the number of slots slightly decreases the point to point bandwidth but increases the number of pairs of Nodes that may simultaneously achieve the maximum rate.

If  $m > 1$  Nodes all wish to transmit simultaneously then each Node is still guaranteed access to one in every  $m + n$  slots (provided of course it is not backed off by a slow receiver). The bandwidth is effectively shared out with each Node achieving at least  $4/(m + n)$  Mbps unless limited by the maximum value.

## 5.2 Cambridge Ring Simulation Outline

### 5.2.1 The Ring

The Ring consists of a number (N) of nodes linked together with the Monitor on a ring, whose length can be specified in terms of a number of bits (R). In this simulation the nodes and Monitor are equally spaced around the ring, so that the delay between successive nodes  $Y(x)$  is given by

$$Y(x) = R / ((N + 1) * F) \text{ microseconds}$$

where  $F$  is the clock frequency in MHz.

A number (ISLOT) of slots of length S bits circulate round the ring in a contiguous group followed by a Gap of length LG, which fills the remaining space on the ring. In this simulation  $S = 40$  and  $ISLOTS = INT (R/S)$  so that the gap can never be as long as S.

On initialisation of the simulation phase the slots are generated in turn and scheduled to arrive at node 1 then the Gap is generated and scheduled to arrive at node 1. Thereafter each slot (and the Gap) is scheduled to arrive at its next destination node.

#### 5.2.2 The Node Structure

Each node is considered to consist of two processors:

- (a) a Host Processor which performs the functions of the Station, and
- (b) a Communication Processor which performs the functions of a repeater.

The Host Processor generates a series of minipackets. The generation of the minipackets depends on:

- (1) GENTT(I), the generation of the next packet at node I, and
- (2) IHTOCQ(I), the room left in the Host-to-Communication queue, if  $GENT > \text{Current Time}$  or  $IHTOCQ = 0$  no minipacket can be generated.

At present a queue length of 1 has been used exclusively. On generation the minipacket is scheduled to arrive at the Communication Processor after a delay of  $\text{Length}/\text{HOSTS}$ , where HOSTS is the Host Processor Speed, and IHTOCQ is reduced by 1.

On arrival at the Communication Processor the Output Buffer (JB) is set to the minipackets index signifying a full Output Buffer and the minipacket is rescheduled to SIMT (effectively suspending the minipacket). When an empty slot arrives the packet index is copied into the slots IDTX field and the Packet Sent Flag (JF) is set to 1.

On the arrival of the slot at its destination the minipacket is received provided that

- (1) the Input Buffer is empty (= 0) and
- (2) the Source Select Switch (IR) = 255  
or the value of the minipackets source,

then a copy of the minipacket is produced and its index copied into the Input Buffer, the new copy is scheduled to arrive at the destination Host Processor after a delay of  $\text{Length}/\text{COMMS}$ , where COMMS is the speed of the Communication Processor. Finally the original minipackets Acknowledgement field (IACK) is set to indicate its return state

IACK = 0 minipacket successfully received  
= 1 destination node busy (IB <> 0)  
= 2 Source Select set for other node  
= 3 destination not receiving (IR = 0)

On the slot's return to the source node the slot's IDTX field is set to -1 (empty) and the JB and JF fields of the source are set to 0 (empty). If the minipacket has been received it is deleted from the lists, otherwise the number of unsuccessful round trips (NOHOPS) is incremented and the minipacket scheduled to reactivate after a number of ring cycles (1 if NOHOPS = 1 else 15).

### 5.2.3 Transactions

All data transfer is by means of transactions (TX's) which have many associated fields, e.g. SCDTIM (Scheduled time of next action), SYSENT (System entry time) etc.

The indices of all TX's are stored in one of two linked lists. These are:

- (1) Available TX's: TX's which are not in the system
- (2) Active TX's: TX's which are active in some way  
(including temporarily suspended).

The headers of these two lists are NEXTAV and NEXTAC respectively. Both lists are linked using the NEXTTX field of the transactions.



There are considered to be five types of active TX in the system. These together with their differentiating field values are tabulated below:

TX Type	IDTX	IDSRC	IDDEST	IDCTND
	Identity	Source Node	Destination Node	Current Node
GAP	0	1	j	0
EMPTY SLOT	-1	1	j	0
FULL SLOT	x	1	j	0
m-p at source	x	1	j	i
M-p at dest.	x	1	j	j

#### The Gap

The Gap is a TX whose IDTX is 0 and whose IDCTND is 0. Its length is the difference between the Ring Size (IRCMD) and the sum of the lengths of the slots. Its only action is to continually circle the ring.

#### Empty Slots

If at its destination node output buffer is full and its sent flag is not set then the output buffer is copied into the empty slot id field and the sent flag is set. In any case the slot is scheduled for arrival at the next node.

### Full Slots

Unless its destination node is either the destination or source of the contained packet the slot steps round the ring.

If the destination node is also the destination of the packet then provided that Input Buffer (IB) = 0 and Source Select (IR) = 255 or IR = packet source, then a copy of the packet is produced and its (the copy's) index is written into the Input Buffer and IACK is set to 0.

Otherwise IACK is set to the relevant value.

If the destination is the source of the contained packet then the IDTX field of the slot is set to -1 (empty) and the Output Buffer (JB) and Sent Flag (JF) are cleared (set to 0).

After either of these cases the slot steps round the ring.

### m-p at source

On generation, the minipacket is scheduled to arrive at the Communication Processor after a delay of Length/HOSTS. On arrival at the Communication Processor the Output Buffer (JB) is filled with the minipackets index and the minipacket is suspended by scheduling its next action at SIMT. When the slot which carried this packet returns, if it has been successfully received it is deleted from the system, otherwise its

NOHOPS field (number of unsuccessful round trips) is incremented and it is rescheduled with a delay of either 1 or 15 ring cycles depending on whether NOHOPS = 1 or not.

#### m-p at destination

The fields of the original minipacket are copied into those of a new minipacket except for IDCTND and SCDTIM, then the new packet is scheduled to arrive at the Host after Length/COMMS. On arrival at the Host, statistics are gathered and the packet is deleted.

#### 5.2.7 Reading and Writing Timings

After a minipacket is generated at the host it is delayed by LENGTH/HOST SPEED before being written into the output buffer. Careful recoding of report may give better figures (as presently this delay is 0.4  $\mu$ seconds. All runs with HOSTSpeed of 100 Mbits/sec. and minipacket of 40 bits length.)

At the destination a delay from IB to host reception of LENGTH/COMM Speed is implemented this ranges from 0.4  $\mu$ secs (COMMS = 100 Mbit/sec.) to 40  $\mu$ sec (COMMS = 1 Mbit/sec).

#### 5.2.5 Future Developments

A new type of TX could be used which would represent a block of data arriving at the Host Processor for transmission.

This new TX type could be identified by setting IDCTND>255 and storing its INDEX in say IHTOC. Also schedule it till the SIMULATION END in CALL SCHEDU (TX INDEX, SIMT). Then generate minipackets at the Host and copy their destinations from the block and decrease the length of the Block TX by the length of a minipacket when the first minipacket is returned acknowledged delete the Block TX.

Note at each simulation increment it will be necessary to go round each node and reschedule any Block TX till the new SIMT.

This is done at present when the minipacket reaches its destination. It may be advisable to do this when the minipacket returns to its source Acknowledged.

### 5.3 Some Simulation Results

In Table 1 results of a number of simulation runs are tabulated.

# CAMBRIDGE RING RESULTS

## Experimental Constants

Clock Frequency	10 MHz
Host Processing Speed	100 Mbits/sec
Simulation Time	0.1 seconds
Average Inter Arrival Time	1.0 micro seconds per minipacket
Arrival Time Distribution	Exponential

## Results

### Average Response Time (micro seconds)

Ring Size (bits)	No. Slots	Number of Ring Nodes*				
		5	10	20	50	100
45	1	25	48	93	227	452
55		31	58	113	277	553
65		36	69	134	329	652
75		42	79	154	379	751
85	2	26	47	77	206	416
95		29	53	100	242	479
105		32	58	111	268	529
115		35	64	121	293	580
125	3	28	48	90	215	423
135		30	52	97	232	456
145		32	56	104	249	490
155		34	54	106	266	523
165	4	29	50	87	204	416
175		31	53	97	213	445
185		33	56	102	241	455
195		35	59	108	254	497
205	5	31	52	93	216	420

\* Note that in this simulation the number of nodes has no effect on the ring size

# REFERENCES

- [1] PAKER, Y., "Control, Synchronization and Reconfigurability problems in Variable Topology Multicomputer Systems", Technical Report, European Research Office, US Army, Grant DA-ERO-78-G-110, 1980
- [2] PAKER, Y., "Variable Topology Multicomputer System", Technical Report, European Research Office, US Army, Grant DA-ERO-124-74-50079, 1975
- [3] PAKER, Y., "Variable Topology Multicomputer System Evaluation", Technical Report, European Research Office, Grant DAJA 37-35-0401, 1976
- [4] PAKER, Y., "Evaluation of Interconnection Topologies by Distributed Computer Modelling Package - MICROSS", Technical Report, European Research Office, US Army, Grant DA-ERO-78-G-110, 1983
- [5] SUMNEY, L.W., "VSLI with a Vengeance", IEEE Spectrum, April 1980, pp.24-27
- [6] PAKER, Y., "Multi-Microprocessor Systems", Academic Press, 1983
- [7] METCALFE, R.M., BOGGS, D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks", Comm. of ACM, Vol.19, 1976, pp.395-404
- [8] HOPPER, A., "Local Area Computer Communication Network", Ph.D. Thesis, Cambridge University, Technical Report 7, April 1978
- [9] MAURELLO, R., "A Distributed Processing System for Military Applications - Part 2; Serial data bus", Computer Design, October 1980, pp.14-36

# REFERENCES

- [10] PAKER, Y. and VERJUS, J.P., Ed., "Distributed Computer Systems", Academic Press, 1983
- [11] TEMPLE, S., "The Design of the Cambridge Fast Ring", Ring Technology Local Area Networks, Eds. I.N. Dallas and E.B. Spratt, Newton Holland, 1984
- [12] ABRAMSON, N., "The Aloha System - another alternative for communications", Proc. of the 1970 Fall Joint Computer Conference. Vol. 37, AFIPS Press, Montvale, N.J. 1970, pp.281-285
- [13] CARLEIAL, A.B. and HELLMAN, M.E., "Bistable Behaviour of ALOHA-Type Systems", IEEE Trans. on Communications, Vol. Com. 23, No.4, April 1975
- [14] KLEINROCK, L. and LAM, S.S., "Packet Switching in a Multi-Access Broadcast Channel: Performance Evaluation", IEEE Trans. on Comm. Vol. Com 23, No.4, April 1975
- [15] TOBAGI, F. and HUNT, V., "Performance Analysis of Carrier Sense Multiple Access with Collision Detection", Computer Networks, Vol.4, 1980, pp.245-259
- [16] LAM, S.S., "A Carrier Sense Multiple Access Protocol for Local Networks", Computer Networks, Vol.4, February 1980, pp.21-32
- [17] METCALFE, J.R. and BOGGS, D., "Ethernet: Distributed Switching for Local Computer Networks", Communications of the Association for Computing Machinery, Vol.19, 1976, pp.395-404

# REFERENCES

- [18] MEDITCH, J. and LEA, C., "Stability and Optimization of the CSMA and CSMA/CD Channels", IEEE Trans. on Comm. Vol. Com 31, No.6, June 1983
- [19] SHOCH, J. and HUPP, J., "Measured Performance of an Ethernet Local Network", Communications of the Association for Computing Machinery, Vol.23, December 1980, pp.711-721
- [20] "The Evolution of Ethernet", Computer Magazine, IEEE Computer Society, August 1982
- [21] IEEE Standard 802.3, The Institute of Electrical and Electronics Engineers, Inc.
- [22] PAKER, Y., "MICROSS: Distributed Computer Modelling Package", PCL Report, June 1983
- [23] PAKER, Y., ENGLISH, H. and BOZYIGIT, M., "NPL-Multicomputer Ring Modelling", PCL Report, February 1983
- [24] BRADY, P.T., "A Statistical Analysis of On-Off Patterns in 16 Conversations Bell System", Technical Journal, January 1968
- [25] KLEINROCK, L. "Queueing Systems, Vol.1 - Theory", John Wiley & Sons Inc., 1975, p.17
- [26] BUX, W., "Local-Area Subnetworks; A Performance Comparison", IEEE Trans. on Comm., Vol. COM-29, No.10, October 1981
- [27] RABINER, L.R. and SCHAFER, R.W., "Digital Processing of Speech Signals", Prentice Hall, 1978



# REFERENCES

- [28] SCHWARTZ, M., "Information Transmission, Modulation and Noise", 3rd ed., McGraw-Hill, 1980
- [29] CHOCHIERE, R.E. and FLANAGAN, J.L., "Current Perspectives on Digital Speech", IEEE Communications Magazine, January 1983, p.32
- [30] BRADY, P.T., "A Statistical Analysis of On-Off Patterns in 16 Conversations", Bell System Technical Journal, January 1968
- [31] "Hatch, Models for the Subjective Effects of Loss, Noise and Talker Echo on Telephone Connections", Bell System Technical Journal, Vol.55, November 1976
- [32] BRADY, P.T., "Effects of Transmission Delay on Conversational Behaviour on Echo-Free Telephone Circuits", Bell System Technical Journal, Vol.50, January 1971
- [33] SWINEHART, D.C. et al., "Adding Voice to an Office Computer Network", Proc. Globe Com, 1983
- [34] TOBAGI, F.A. and GONZALEZ-CRAWLEY, N., "On CSMA/CD Local Networks on Voice Communications, Proc. InfoCom 1982, pp.122-127
- [35] JOHNSON, D.H. and O'LEARY, G.C., "A Local Access Network for Packetized Digital Voice Communication", IEEE Trans. on Comm., Vol. COM-29, No.5, May 1981
- [36] GONSALVES, T.A., "Packet Voice Communication on an Ethernet Local Computer Network: an Experimental Study", Proc. SIGCOMM 1983, Symposium on Comm. Arch and Protocols, Austin TX, March 1983, pp.178-185

#### REFERENCES

- [37] SHOCH, J.F., "Carrying Voice Traffic Through an Ethernet Local Network - A General Overview in Local Networks for Computer Communications", A. West, P. Janson (editors), North Holland Publishing Co., IFIP, 1981
  
- [38] Cambridge Ring 82 Protocol Specifications. Joint Network Team of the Computer Board and Research Councils, November 1982
  
- [39] Cambridge Ring 82 Interface Specifications. Joint Network Team of the Computer Board and Research Councils, September 1982

Appendix 1: A sample run for CSMA/CD Contention Bus Simulator

```
INPUT SELECT NEXT OPTION
  1 = DEFINE SYSTEM
  2 = SIMULATE
  3 = REPORT
  4 = RESTART
  5 = SNAPSHOT
  6 = STOP
MONITR>1
SELECT HARDWARE
  ENTER CHANNEL CAPACITY(MEGABITS/SEC)
    AND CABLE LENGTH(KILOMETERS)
10 2
SELECT PROTOCOL SPECIFICATION
  ETHERNET: 0
  CSMA/CD : 1
0
SELECT COMMUNICATIONS TRAFFIC SOURCES
(MAXIMUM OF 256 NODES)
  INPUT NUMBER OF DATA NODES
8
  INPUT NUMBER OF VOICE NODES
0
  INPUT AVERAGE INTER-ARRIVAL TIME (SECS)
  FOLLOWED BY DISTRIBUTION
  FOR THE DATA SOURCES
  DISTRIBUTION  1 = FIXED
                2 = EXPONENTIAL
0.004 2
  INPUT AVERAGE PACKET DATA LENGTH (BITS)
  FOLLOWED BY DISTRIBUTION
  FOR THE DATA SOURCES
  DISTRIBUTION  1 = FIXED
                2 = EXPONENTIAL
1000 2
INPUT SELECT NEXT OPTION
  1 = DEFINE SYSTEM
  2 = SIMULATE
  3 = REPORT
  4 = RESTART
  5 = SNAPSHOT
  6 = STOP
MONITR>2
ETHSIM: INPUT SIMULATION DURATION
10
```

INPUT SELECT NEXT OPTION

- 1 = DEFINE SYSTEM
- 2 = SIMULATE
- 3 = REPORT
- 4 = RESTART
- 5 = SNAPSHOT
- 6 = STOP

MONITR>3

REPORT:

QUEUEING DELAY= 0.09952048(MILISEC/PACKET)  
 QUEUEING + TRANSMISSION  
 DELAY= 0.25662048(MILISEC/PACKET)  
 THROUGHPUT= 3.1508744955(MEGABITS/SEC)  
 % OF PACKETS LOST = 0  
 % OF PACKETS SENT = 20045  
 % OF BITS SENT = 31508745

SERVICE OF PACKETS LAGS ARRIVAL PROCESS

DO YOU WISH TO SEE TABLE (1 = YES)

1

NODE= 1 LAG= 0.00000(MILISEC)  
 NODE= 2 LAG= 0.00000(MILISEC)  
 NODE= 3 LAG= 0.00000(MILISEC)  
 NODE= 4 LAG= 0.00000(MILISEC)  
 NODE= 5 LAG= 0.00000(MILISEC)  
 NODE= 6 LAG= 0.00000(MILISEC)  
 NODE= 7 LAG= 0.00000(MILISEC)  
 NODE= 8 LAG= 0.00000(MILISEC)

PACKET QUEUEING AND XMISSION DELAY HISTOGRAM

DO YOU WISH TO SEE IT

0 = NO

1 = WHOLE THING

2 <CR> X ONLY UP TO X, LIMITED 1-256(MILISEC)

2

10

FIRST PERCENTAGE IS

DISTRIBUTION OF QUEUEING DELAY

SECOND PERCENTAGE IS

DISTRIBUTION OF QUEUEING + XMISSION DELAY

RANGE (MILISEC)	% CUM	% CUM
0.TO 1.	98.16	97.53
0.TO 2.	99.53	99.44
0.TO 3.	99.83	99.81
0.TO 4.	99.88	99.88
0.TO 5.	99.92	99.91
0.TO 6.	99.95	99.95
0.TO 7.	99.99	99.97
0.TO 8.	99.99	99.98
0.TO 9.	99.99	99.99
0.TO 10.	100.00	100.00

BREAKDOWN OF # OF PACKET COLLISIONS BY DELAY  
DO YOU WISH TO SEE IT (1 = YES)

RANGE (MILISEC)	TOTAL NUMBER	0 COL	1 COL	2 COL	3 COL	4 COL	5 COL	6 COL	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.TC 1.	19676.	16954.	1233.	935.	462.	89.	3.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
1.TC 2.	275.	31.	15.	7.	48.	140.	32.	2.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
2.TC 3.	60.	13.	2.	2.	4.	4.	29.	6.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
3.TC 4.	10.	2.	1.	0.	1.	0.	1.	5.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
4.TC 5.	8.	1.	0.	0.	0.	0.	0.	7.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
5.TC 6.	6.	3.	0.	0.	1.	0.	0.	2.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
6.TC 7.	5.	1.	0.	0.	0.	0.	0.	0.	4.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
7.TC 8.	2.	1.	0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
8.TC 9.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
9.TC 10.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
12.TC 13.	1.	0.	0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TOTAL NUMBER		17006.	1251.	944.	516.	233.	65.	22.	7.	1.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.
NUMBER OF ATTEMPTS		17006.	2502.	2832.	2064.	1165.	390.	154.	56.	9.	9.	0.	0.	0.	0.	0.	0.	0.	0.	0.

ATTEMPTS TO GAIN CHANNEL= 2.6177999973(KILO/SEC)  
V. ATTEMPTS PER PACKET= 1.3059615791

## Appendix 2: Simulation Variables mentioned in the text

### Subroutines

RAN	(Fortran library) a pseudo random real number generator
IRAND	a pseudo random integer generator
SCHEDU	changes the time when events are scheduled for transmission by modifying the event queue
GENERA	uses random numbers to create events which it schedules on the event queue
BBOSCH	implements the binary backoff algorithm to delay an event, and schedules it on the event queue.

### Common blocks

RNGHWR	contains global variables concerning system hardware
PACKET	contains global variables concerning the event queue

### Variables (see routine ETHDEF)

NEXTAC	pointer to list of active data structures - the top of the event queue
NEXTAU	pointer to list of unused data structures
SYSENT	arrival time of packet at a station
SCDTIM	scheduled time for packet transmission attempt, originally SYSENT, but modified if packet is deferred or collides. This value orders the event queue.
SIMT	total simulation time
TIME	current time within the simulator
SIMINC	incremental time step - smallest time resolution
IRSWCH	state of simulator

**END**

**FILMED**

**11-84**

**DTIC**